Disk Magazine

Strutture dati

Capire e utilizzare il C

A8600 l'Europa videounita

3D in Basic

Spedizione in abbonamento postale Gruppo III/70

EDITORIALE

La porta sul mondo Amiga

Un'entrata, senza ritorno, nell'atmosfera e nell' ambiente più effervescente del momento. Da qui l'impegno a dispiegarne l'intreccio e le possibilità. A riportarne gli stimoli ed i fermenti, senza privarci del fantasmagorico e dello spettacolare. Ma tutto all'insegna della più raffinata e documentata informazione. Questi, a grandi linee, gli scopi di questa rivista. È per questo motivo che nelle varie sezioni troveranno posto, volta per volta, diversi linguaggi: Assembler, Basic, C, Modula-2, Forth, Lisp, ecc. Nel tentativo di conservare le tradizionali tendenze programmative e, contemporaneamente, assecondare quelle che vanno sempre più affermandosi.

Un progetto, dunque, organico ed integrato, che non vuole tralasciare nulla della ricchezza e della complessità dell'universo informatico in cui intendiamo addentrarci. E la simbiosi rivista-disco realizza in modo insostituibile questo nostro obiettivo, nel senso che pur conservando una rispettiva autonomia, è la loro interrelazione che produce quell'unità e completezza d'informazione che tanto ci preme. Infatti, i contenuti del disco vengono supportati adeguatamente con esaurienti spiegazioni nelle varie sezioni all'interno della rivista e nelle otto pagine centrali che costituiscono l'inserto dedicato interamente al disco.

Ma non basta. In relazione ai diversi linguaggi, si è potuto, grazie a questo duplice supporto, rendere ancor più fruibili i diversi contenuti, sostenendo lo sforzo di un adeguato taglio espositivo, dal didattico-introduttivo allo strettamente tecnico, senza timore di mettere in difficoltà l'utente per l'editazione o per la compilazione dei programmi presentati, dal momento che i sorgenti e i codici oggetto trovano posto nelle appropriate directory del disco. In questo modo abbiamo salvaguardato il diritto ad una fruizione progressiva del materiale propria di coloro che incominciano l'avventura della programmazione, senza però togliere nulla alla completezza e all'approfondimento che interessano maggiormente gli iniziati e gli smaliziati.

Entrare, comunque, da protagonisti in questo universo. E questa è la porta sul mondo Amiga: she's magic.

La Redazione

SOMMARIO

Foto di copertina tratta da:
«Grafica 3D in tempo reale» *di Paolo Russo*

Anno I numero 1 Luglio/Agosto 1988

DIRETTORE RESPONSABILE
Giampietro Zanga

REDAZIONE
Graphic & Comp. Gorizia

COORDINAMENTO REDAZIONALE
Simone Concina

ART DIRECTOR
Gianni Marega

COLLABORATORI
Roberto Beccia
Primoz Beltram
Tomi Beltram
Fabio Biancotto
Giorgio Dose
Mr. Lambda
Massimo Lavarin
Furio Lusnig
Luigi Manzo
Giovanni Michelon
Emilio Orione
Alessandro Prandi
Paolo Russo

GRAFICA, IMPAGINAZIONE, COPERTINA
Graphic & Comp.

DIVISIONE PUBBLICITÀ
Via Pola, 9 - 20124 MILANO - Tel. 69.481
Telex 316213 REINAI - 333436 GEJ - ITI
OVERSEAS DEPARTMENT: Tel. 02/6948201
PUBBLICITÀ PER ROMA-LAZIO E CENTRO SUD
Via Lago di Tana, 16 - 00199 Roma
Tel. (06) 8380547 - Telefax (06) 8380637

FOTOCOMPOSIZIONE
FOTOFORMA - Via del Molino a Vento, 72
34137 TRIESTE

STAMPA
Grafika, 78 - Pioltello

DISTRIBUZIONE
Sodip - Via Zuretti, 25 - 20125 MILANO
Spedizione in abbonamento postale Gruppo III/70
Pubblicità inferiore al 70%

UFFICIO ABBONAMENTI
Tel. (02) 6122527-6187376
Prezzo della rivista L. 14.000 (Frs.21.00)
Numero arretrato L. 28.000
Abbonamento annuo L. 135.000
per l'Estero L. 270.000
I versamenti vanno indirizzati a:
Gruppo Editoriale Jackson
Via Rosellini, 12 - 20124 Milano
mediante emissione di assegno bancario, vaglia o utilizzando il C/C/ postale numero 11666203
Per i cambi di indirizzo, indicare, oltre al nuovo, anche l'indirizzo precedente, ed allegare L.500, anche in francobolli.

DIVISIONE PERIODICI
DIREZIONE, REDAZIONE, AMMINISTRAZIONE
Via Rossellini, 12 - 20124 Milano
Tel. (02) 68.80.951/2/3/4/5 - Telex 333436 GEJIT I

SEDE LEGALE
Via G. Pozzone, 5 - 20121 Milano

Il Gruppo Editoriale Jackson è iscritto nel Registro nazionale della Stampa al n. 117 vol. 2 - foglio 129 in data 17/8/1982

Autorizzazione del Tribunale di Milano n. 102 del 22/2/1988

Galileo

La luce ed il chiarore delle stelle hanno sempre esercitato una notevole influenza sull'uomo. ...e questa sera la prima stella su Amiga...

GALILEO è senza dubbio il miglior programma di astronomia prodotto per computer al giorno d'oggi a disposizione. Utilizzando la grafica e la potenza del computo messe a disposizione dall'Amiga, è capace di visualizzare il cielo notturno come appare nella realtà da una qualsiasi posizione della Terra in qualsiasi momento e per un periodo della lunghezza di 400 anni! Il programma non è soltanto un ottimo tool per l'osservazione astronomica; ma è anche utilissimo per l'apprendimento dei concetti che sono basilari per l'osservazione astronomica. Il programma è interamente guidato da dei menù, con i quali si possono individuare le posizioni di 1600 stelle, pianeti e altri oggetti come galassie, nebulose, ecc.

È in grado di utilizzare per l'analisi dei dati: o il sistema di coordinate equatoriali oppure quello relativo all'altitudine e all'azimut. Galileo, dopo che avrete immesso la latitudine, la longitudine e la data visualizzerà gli oggetti celesti che avrete richiesto con l'angolazione da voi fornita. La funzione di plotting è lenta se comparata ad altri programmi di grafica realizzati con l'Amiga, ma è sorprendentemente veloce se messa in relazione con altri programmi di astronomia. I calcoli per creare il cielo notturno sono molto elaborati e richiedono perciò un determinato lasso di tempo.

Il programma è fornito di molte opzioni interessanti. È possibile visualizzare il percorso effettuato dai pianeti avendo per riferimento le stelle. Vengono forniti su richiesta i nomi dei pianeti, delle stelle, delle costellazioni, e alcuni oggetti NGC. Si possono visualizzare le costellazioni in modo che si possano riconoscere poi anche nella realtà. Altre opzioni vi permetteranno la ricerca di pianeti e costellazioni in modo da ottenere una magnifica visione dei pianeti. La funzione 'What's Up' vi indica quali pianeti sono visibili nel cielo al mattino ed alla sera nella posizione da voi richiesta.

Il programma impressiona favorevolmente anche per la cura riposta nel determinare i dettagli, ma comunque non e' ancora all'altezza di rimpiazzare un buon atlante astronomico. Galileo è utile per apprendere il sistema di lettura che si serve delle coordinate e per indagare sulla posizione delle stelle e dei pianeti in date particolari. Se il vostro interesse nei riguardi dell'astronomia è più di tipo educativo che scientifico, allora GALILEO e Il programma che fa per voi.

Infinity Software Ltd., 1331 61st Street, Suite F Emeryville, CA 94608, USA.

Gold Spell

Siete soliti scrivere con una mano sul dizionario e con l'altra sulla tastiera? Se qualcuno nel chiedervi di fare lo spelling della parola 'partenogenesi' vi fa sudare freddo; allora questo programma è adatto a voi...

Gold Spell è esattamente ciò che immaginate, verifica l'esatta ortografia delle parole. Contiene oltre 90.000 parole, è compatibile con Textcraft, Scribble, o un qualsiasi word processor per Amiga che registri i file in formato ASCII (solo testo), permette inoltre di aggiungere vocaboli al dizionario di cui dispone.

Il programma è di facile utilizzo, basterà caricarlo in memoria e fornirgli il nome del file che si desidera testare e immediatamente inizierà a verificare il documento. Se trova una parola che non riconosce, si arresta, visualizza l'intera frase ed evidenzia la parola in oggetto. Voi potrete correggere la parola, accettare la parola, accettare e far ricordare al programma la parola (utile per aggiungere vocaboli al dizionario), chiedere al programma di suggerirvi la parola corretta, o analizzare il dizionario per trovare la parola corretta. Dopo avere trovato la parola esatta, tutto quello che dovete fare è premere il pulsante del mouse e il vocabolo verrà inserito automaticamente nel testo. Quando il documento è stato completamente analizzato, il programma registra la versione corretta utilizzando il nome originale e simultaneamente registra la vecchia versione aggiungendo posteriormente al nome del file la seguente dicitura '.BAK'. A questo punto potete aggiornare il vocabolario con le nuove parole individuate.

Ci sono inoltre alcune interessanti configurazioni nel programma che vanno oltre l'esatta scrittura dei termini; per esempio e' possibile analizzare la leggibilità di un documento. Il vostro dizionario aumenterà con l'inserimento di nuove parole ed avrà come limite di capacità l'ampiezza della memoria del vostro Amiga. Gold Spell è molto veloce poiché il dizionario è posto nella RAM. Il programma è veramente eccellente ed è venduto ad un ottimo prezzo ($45.95).

PO Box 789 Streetsville, Mississauga, Ontario L5M 2C2, Canada.

Apprendimento e computer

Se i vostri figli necessitano di una ripassata scolastica durante l'estate, La MicroEd ha realizzato vari pacchetti educazionali che si vanno ad aggiungere alla loro produzione già più che soddisfacente: Beginning Counting al prezzo di $39.95 (due dischi), Making Our Constitution al prezzo di $79.95 (quattro dischi), Transcontinental Railroad al prezzo di $39.95 e due volumi di Learning American English al prezzo di $89.95 (cinque dischi per volume).

I programmi utilizzano lo stesso tipo di approccio utilizzato nei programmi precedenti già prodotti dalla MicroEd: sezioni riservate per le domande e le risposte, quiz e digitazione di disegni. Il programma privilegia la comprensione di tipo uditivo piuttosto che dar risalto ad una pronuncia corretta. Tutti i programmi presentati utilizzano la voce dell'Amiga.

MicroEd Incorporation, PO Box 24750, Edina, MN 55424, USA.

Programmi educativi

La Other Guys propone dei programmi di apprendimento scolastico basati fondalmentalmente sul gioco, utilizzando grafici, musica e parola. Match-It viene venduto al prezzo di $39.95 ed insegna i colori e le forme fondamentali. Math a Magician venduto al prezzo di $39.95 ricopre il campo delle quattro operazioni con numeri interi e frazionari a quattro livelli di difficoltà.

Talking Storybook venduto al prezzo di $49.95, legge delle storie utilizzando fino a dieci voci diverse per storia, è possibile spegnere la voce e leggere o far leggere al bimbo il testo visualizzato sullo schermo. Storie addizionali vengono vendute ad un prezzo che oscilla tra i 24$ e i 39.95$. Promise è un programma per la verifica dello spelling e viene venduto al prezzo di $49.95.

The Other Guys, 55 North Main Street, Suite 301-D, PO Box H, Logan, UT 84321, USA.

La rubrica "Corrispondenza"
è stata voluta
al fine di definire uno spazio,
interno alla rivista, dedicato al confronto
e al contributo di idee, provenienti
dalla galassia dell'utenza Amiga.
Per avervi accesso,
inviate le vostre missive a:

Spett. redazione
"Amiga Magazine"
rubrica "Corrispondenza"
Gruppo Editoriale Jackson
via Rosellini, 12
20124 MILANO

AMIGA TRICKS

Cercasi comando

Ci sono quattro comandi dell'AmigaDOS che non sono presenti sul disco Workbenk. Questi sono ALINK, ASSEM, DOWNLOAD e READ. Questi comandi fanno parte dell'AmigaDOS, ma non sono inclusi nel disco che accompagna il vostro Amiga. Questi sono dei comandi di rilevazione. Essi sono inclusi in sistemi di linguaggio third-party.

La versione 1.2 del sistema operativo contiene anch'essa alcuni comandi nuovi dell'AmigaDOS.

Multiuso

Una delle configurazioni più utili dell'Amiga è il multitasking, ma sfortunatamente non è una macchina multiuso. C'è, comunque, una configurazione poco conosciuta che permette di essere utilizzata nel modo sopra accennato. Quando si è in una window CLI, si digiti:

```
NEWCLI SER:
```

Quest'operazione attiverà l'apertura di un nuovo processo CLI verso la porta seriale. Quest'operazione vi permette di agganciare un terminal esterno per far operare i programmi CLI sullo sfondo. Per concludere il processo del CLI, basterà digitare:

```
ENDCLI SER:
```

e premere varie volte il tasto Return per spazzare il buffer. Sfortunatamente, non conosciamo alcun mezzo per evitare il buffer tra il terminale ed il computer. Ci è pervenuta la notizia che il comando

```
NEWCLI SER:RAW
```

compia quest'operazione, ma noi non l'abbiamo ancora testata e non siamo perciò in grado di dirvi se funziona in modo corretto.

Set di caratteri alternativo

Quasi tutte le persone che hanno giocato con il CLI hanno visualizzato sullo schermo prima o dopo il set di caratteri alternativo. Se si preme un tasto errato o si prova qualcosa di nuovo ecco apparire questi strani caratteri al posto del normale set di caratteri. È successo anche a noi e continuando a premere dei tasti a caso abbiamo rilevato che ci sono due tasti che sono legati a questa anomalia e che perciò possono risultare estremamente utili.

Premendo 'CTRL O' si ritorna al normale set di caratteri e premendo 'CTRL N'ci si posiziona nuovamente sul set alternativo di caratteri. Cosi' se vi accade di veder riempito il vostro schermo da caratteri anomali, non dovrete far altro che premere i tasti 'CTRL O' ritornando cosi' immediatamente al vostro lavoro.

Dentro il Textcraft

Alcuni software, come il Textcraft, non forniscono alcuna spiegazione su come ci si può spostare tra schermi premendo semplicemente alcuni tasti. Ad esempio, con Textcraft, molte persone selezionano l'opzione Quit dal menù quando desiderano visualizzare lo schermo Workbenk per eseguire altre operazioni.

Ma esiste un modo molto più semplice e veloce per passare al Workbenk senza uscire dal textcraft: Aprite semplicemente il Workbenk Clock prima di aprire il Textcraft. Poi, dall'interno del Textcraft, la pressione contemporanea dei tasti 'Amiga N'vi spostera' allo schermo del Workbenk e la pressione dei tasti 'Amiga M' vi riportera' al Textcraft.

Leggere il Joystick in C

Il manuale propone una procedura molto complessa per leggere la porta del joystick in C, così ve ne proponiamo una molto più semplice. Nella routine, ci si è calati all'interno della macchina leggendo direttamente i registri hardware. Questa routine lavora ugualmente bene con i compilatori Lattice e Aztec:

```
#= define PORT 2
short*joy = 0xdff008 + 2 * port;
char*cia = 0xbfe001;

main()[
  for(;;)[
    if(*joy&2)
      printf("RIGHT...");
     if(*joy&512)
      printf("LEFT...");
    if((*joy> > 1^*joy)&1)
      printf("DOWN...");
    if((*joy> > 1^*joy)&256)
      printf("UP...");
    if(!(*cia&64*PORT))
      printf("FIRE!!!");
    printf("/n");
    ]
]
```

Il Comando Paint

Operando con il comando Paint a volte accade di vedersi riempire lo schermo con il colore con cui si voleva riempire una determinata figura come nell'esempio seguente:

```
SCREEN 4,320,200,4,1
WINDOW3,"paint.test",,31,4
maxcolor=15
FOR color.id = 0 TO maxcolor
  CIRCLE(150,100),100
  PAINT(150,100),color.id
  NEXT color.id
SCREEN CLOSE 4
```

Il problema è che il manuale Basic dell'Amiga non è molto chiaro riguardo alla funzione del comando PAINT, inducendo così in errore anche i programmatori più esperti. Il comando ha la seguente sintassi:

```
PAINT[STEP](x,y)[,paintColor-id[,borderColor-id]]
```

L'unica cosa che dovete aggiungere al comando PAINT è la locazione di un unico pixel all'interno dell'area che si desidera riempire. Se non si aggiungono i parametri dei colori, essi assumono per default il valore del colore attuale del fondo. Nell'esempio riportato si dovrà aggiungere un 'paintColor-id'utilizzando la variabile 'color.id'. Quando verra' aggiunto questo paint color e non quello del bordo, il bordo assumerà automa-

ticamente il valore del paint color. Esaminiamo ora cosa accade durante un passaggio attraverso il loop.

Si assuma che la variabile del color.id sia uguale a quattro, il paint color sarà di colore arancio. Il programma disegnerà dapprima, un cerchio con il colore attuale del fondo (che è il colore uno -bianco- finché non si immetterà un comando COLOR per modificarlo). Poi il programma esegue il comando PAINT; colorando l'area attorno alla locazione del pixel specificato con il colore arancio. L'operazione continua finché non raggiunge un bordo il cui colore è definito dal parametro 'borderColor-id'.

Nel programma il 'borderColor-id'e' identico al paint color. Il comando PAINT riempira' la superficie di colore arancio finché non incontra un bordo arancio; il cerchio bianco non sortirà perciò alcun effetto. Finché nel programma non verrà immesso un bordo arancio, il comando PAINT riempirà l'intero schermo. Per far funzionare in maniera corretta il programma si dovrà modificarlo specificando l'attuale color.id all'interno del comando CIRCLE:

```
CIRCLE(150,100),100,color.id
```

In questo modo il comando PAINT avrà il bordo sul quale fermarsi.

Ripulire il buffer di tastiera in Basic

La capacità di digitare in testa (to type ahead on ..) all'Amiga è normalmente un'ottima qualità, che sicuramente riuscirà a sconvolgere il comando INKEY$ in un programma BASIC. In situazioni simili la seguente subroutine può essere di notevole aiuto:

```
SUB CLEARKEYS STATIC
FOR X = 1 TO 10
r$ = INKEY$
NEXT X
ENDSUB
```

Per utilizzarla, richiamatela immediatamente prima del comando INKEY$, come nel seguente esempio:

```
CALL CLEARKEYS
WHILE INKEY$ = "": WEND
```

Registrare delle icone fatte su misura

La prima volta che si scopre come utilizzare l'editor di icone dal disco workbenk, ci si cimenterà sicuramente nella produzione di icone per i propri programmi in Basic. Ma se il programma viene corretto e lo si riregistra, le icone disegnate vengono sostituite da quelle standard che sono state create per i programmi in Amiga Basic. Due possono essere le soluzioni per questo problema:

La prima, è quella di preparare un libreria di icone e utilizzarla, quando è necessario, con un Editor di Icone per sostituire le icone standard con le proprie.

L'altra soluzione viene implementata senza uscire dall'Amiga Basic, e vi lascia con una vecchia copia (non modificata) del programma. Il primo passo è quello di portarsi nel modo intermedio e digitare SAVE OLD. Ora siete in possesso di una nuova copia del programma con delle icone standard e una vecchia copia del programma con le icone da voi disegnate. Non vi resta che commutare i programmi, nel modo intermedio, digitando:

```
NAME OLD AS TEMPORARY
NAME nome del vostro programma
AS OLD
NAME TEMPORARY AS nome del
vostro programma
```

Ora avete il nuovo programma con le icone da voi disegnate e il vecchio programma con quelle standard, così potete spostare la copia dove volete, al limite anche gettarla se ciò che avete elaborato funziona perfettamente.

Say dal Basic

Se desiderate utilizzare la funzione (voce) SAY, il metodo migliore è quello di immagazzinare il testo parlato in un file dati sequenziale creato con la funzione built-in ED o con un qualsiasi word processor che permetta dei save in ASCII. Dopo che il file è stato creato, aggiungete le seguenti linee al vostro programma in Amiga Basic:

```
OPEN filename FOR INPUT AS # 1
  REM filename CREATED USING
ED
WHILE NOT EOF(1)
LINE INPUT# 1,A$
SAY TRANSLATE$(A$)
WEND
CLOSE # 1
```

Potete ora ascoltare il testo parlato prima di inserirlo in un vostro programma aprendo una finestra CLI e digitando SAY-X seguito dal nome del file.

Copia file

Un altro modo di copiare un file è per mezzo dell'utilizzo del comando Type in ambiente CLI. Il comando Type visualizzerà i contenuti di un file in ASCII o esadecimale, e questo dipenderà dall'opzione utilizzata. Normalmente il ritorno avviene o su schermo o su stampante. Comunque, abbiamo scoperto che può effettuarsi anche su disco o file, e se il file non viene specificato, ne verrà creato uno. Il formato per ottenere quanto fin qui detto è il seguente:

```
TYPE DF?: filename TO filename
```

Ad esempio, poniamo come ipotesi di avere un file su disco nel drive 1 denominato AmigaWorld e che vogliamo copiarlo su disco nel drive 0 con il nome Mags. In modo CLI, digiteremo la seguente stringa di comando:

```
TYPE DF1:AMIGAWORLD
DF0:MAGS
```

Si noti che l'utilizzo di TO può essere anche omesso. Ora se effettuate una ricerca nella directory, troverete certamente un nuovo file con il nome di MAGS su DF1:.

Stop al rumore

Chiunque abbia comperato un drive esterno per l'Amiga avrà notato che quando la macchina è accesa e uno dei drive è senza dischetto, questi emetterà ogni pochi secondi un noiosissimo rumore. Per eliminare questo disturbo basterà semplicemente inserire nel drive un qualsiasi disco.

Mi rendo conto che per i più questa può essere una notizia ovvia e banale, ciò non toglie che il dovere di una rivista specializzata è anche quello di rivolgersi a coloro che stanno muovendo i primi passi all'interno di un discorso informatico e, quindi, di fornire anche questo tipo di informazioni che potremmo definire "spicciole".

Formattamento

Prima di poter utilizzare un disco è indispensabile intraprendere quella serie di attività che vanno sotto il termine di "formattamento". Per quanto questa sia una delle fasi indispensabili, è sempre preferibile, qualora sia possibile, eseguirla con una certa celerità. Quello che vi proponiamo ora, pertanto, è un mini programma velocissimo che preparerà per voi dei dischi dati. Utilizzando ED, create il seguente file:

```
echo "Formatting Drive DF1:"
format drive df1: name "Empty"
echo "Installing DF1:"
attendere 4 secondi circa
install df1:
echo "Installed copy finished"
```

Una volta digitato questo file, registratelo con il nome di FORMA1 e quando ne avrete bisogno non dovrete far altro che eseguire forma1 (assicuratevi di avere il disco vuoto in DF1) per essere pronto per l'utilizzo. Il programma proposto contribuirà a farvi risparmiare certamente un po' di tempo nella preparazione dei vostri dischi.

Test drive

Chi non ha mai desiderato provare la potenza delle più famose macchine sportive? Test drive della Accolade vi permette la guida simulata di cinque bolidi velocissimi. Potete provare la velocità di una Ferrari Testarossa, udire l'urlo delle gomme prodotto da una partenza bruciante di una Porsche 911 Turbo, far 'sballare' il livello dell'olio di una Lamborghini Countach, volatilizzarvi in una nuvola di fumo alla guida della vostra Corvette o provare la docilità delle manovre con una Lotus Turbo Esprit.

L'ottima grafica, il suono e l'animazione molto realistica vi permetteranno di verificare le caratteristiche di ciascun mezzo. Il programma prevede inoltre, per ciascuna macchina, la visualizzazione di un foglio con le caratteristiche specifiche di ciascun bolide.

Per visualizzare le macchine basterà muovere il joystick (su-/giù), la pressione del pulsante di fuoco permetterà poi la selezione. La guida verrà ostacolata da macchie d'acqua presenti sul tracciato, da autisti della 'domenica' ecc. Dovrete guidare senza causare incidenti con le altre macchine o con i macigni presenti sulla strada, cercate di non farvi multare per eccesso di velocità e soprattutto cercate di non finire fuori strada. Per completare il percorso avete a disposizione cinque possibilità. Vicino allo specchietto retrovisore c'è il rilevatore radar che vi indicherà la presenza di rilevatori radar della Polizia.

Il programma "Test Drive" è prodotto e distribuito in Italia dalla C.T.O. srl via Indipendenza 40 Bologna.

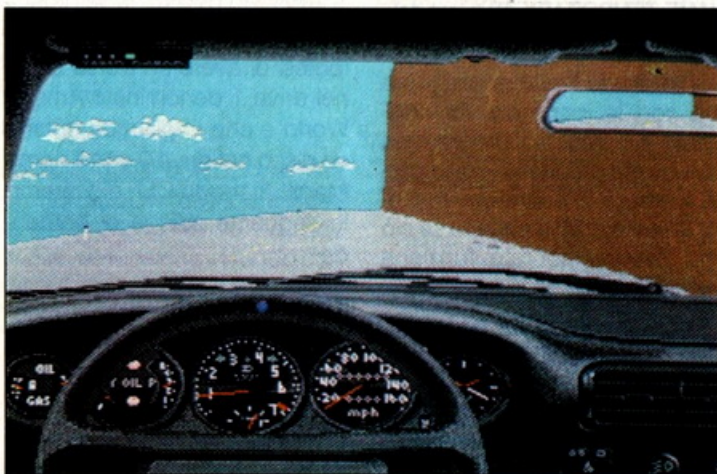
Winter Olympiad 88

Il programma presenta una serie di giochi sportivi invernali (cinque). Tutti i giochi sono realizzati graficamente in modo superbo.

Prima di iniziare il gioco si accede ad una serie di opzioni.

Opzione 1: vi permette di registrare i vostri record personali su disco.

Opzione 2: potete scegliere il numero dei giocatori, da uno a sei.

Opzione 3: immissone del

proprio nome e scelta del paese per cui giocare.

Opzione 4: selezione delle gare.

Il primo gioco è la 'discesa libera' ed è anche il più veloce dei giochi. Dovrete percorrere un tracciato di tre Km. e allo stesso tempo cercare di evitare le rocce ed altri ostacoli. Lo stile in questa disciplina non è rilevante, importante è invece raggiungere l'arrivo nel minor tempo possibile.

Il 'salto con gli sci' trasforma lo sciatore in un uccello e dopo una discesa di 90 metri raggiunge il punto di stacco del trampolino; in questo sport sono importanti per la valutazione sia la distanza coperta che lo stile. È importante durante la fase di volo il ridurre la resistenza all'aria e durante l'atterraggio controllare perfettamente gli sci per portarlo a buon fine.

Lo 'slalom gigante' richiederà all'atleta: agilità e coordinazione per scivolare tra le porte. Una volta trovato il ritmo si dovrà cercare di mantenerlo per portare a buon fine la prova. Il punteggio viene valutato sommando al tempo reale occorso per completare il percorso, il tempo di penalizzazione inflitto per aver 'saltato' eventuali porte.

Il 'bob' e' una delle gare più pericolose delle olimpiadi invernali, in questo sport l'abilità del pilota viene messa a dura prova. Queste macchine assomigliano a dei proiettili che stiano attraversando dei tunnel di ghiaccio.

Il 'biathlon'e' una gara di durata, e' estenuante, sono importanti in questo caso la resistenza e la concentrazione dell'atleta. Quando raggiunge i poligoni di tiro bisogna concentrarsi rapidamente e cercare di non fallire nessun bersaglio in caso contrario si verrà penalizzati. Il punteggio finale viene valutato sommando il tempo occorso per percorrere l'intero tragitto e i tempi di penalizzazione inflitti per i bersagli mancati.

Il programma "Winter Olympiad 88" è prodotto e distribuito in Italia dalla C.T.O. srl via Indipendenza 40 Bologna.

The art of chess

Questa versione tridimensionale degli scacchi è accompagnata da un opuscolo il quale illustra le principali regole degli scacchi e presenta un breve cenno storico delle macchine ideate per giocare agli scacchi.

Il programma è controllato interamente da menù o da accorgimenti visualizzati sullo schermo. Per caricarlo in memoria si premano contemporaneamente i tasti: Ctrl, Commodore e Amiga. Prima di utilizzare il programma vi sarà richiesto di fornire alla macchina una serie di risposte, che altro non è che una sequenza protettiva del programma.

Per visualizzare le opzioni del menù, si dovrà premere il pulsante destro del mouse e a questo punto appariranno sulla parte alta dello schermo le seguenti opzioni:

PROJECT, PLAY, EDIT, SPECIAL, CLOCKS, SOUND.

Ora, mantenendo premuto il pulsante, si sposti la freccia su una di queste opzioni. Selezionando una qualsiasi delle scelte a disposizione accederete a dei sub-menù. Spostando la freccia sulla lista le opzioni verranno evidenziate in reverse, a questo punto per determinare la scelta desiderata basterà non premere più il pulsante di destra del mouse. In alcuni casi verranno visualizzate delle ulteriori sub-liste.

Le prime due opzioni di PROJECT sono:

Start Game with Amiga White (Amiga con pezzi bianchi).

Start Game with Amiga Black (Amiga con pezzi neri).

Per iniziare il gioco basterà scegliere una di queste due voci. Questa selezione sarà disponibile in qualsiasi momento della partita. Se siete a partita inoltrata e desiderate utilizzare questa opzione ricordatevi di registrare la partita, in caso contrario la partita in corso verrà irrimediabilmente persa.

Per muovere un pezzo, ci si posizioni con la freccia sul pezzo scelto e si prema il pulsante di sinistra del mouse. Con il pulsante mantenuto premuto si sposti il pezzo sulla casella scelta, a questo punto basterà rilasciare il pulsante. Se la mossa è corretta il pezzo si posizionerà sulla casella desiderata e l'Amiga inizierà a pensare alla sua risposta; se invece la mossa effettuata è illegale il pezzo ritornerà automaticamente al posto di partenza. Per quanto riguarda l'arrocco basterà semplicemete posizionare il RE sulla casella desiderata e se l'arrocco è corretto, la macchina provvederà a completarne la sequenza. Durante il corso della partita o quando selezionate il menù o il gadget, la linea in alto sullo schermo visualizzerà un messaggio. Tale linea di commento vi indicherà il vostro turno, se siete sotto scacco, ecc.

Sulla parte destra dello schermo, è visualizzata una freccia con doppia punta detta 'Time Travel Gadget' che vi permette di spostarvi, durante una partita, in avanti o indietro in modo da visualizzare le mosse effettuate. Per rivedere delle mosse precedenti ci si posizioni sulla punta della freccia rivolta verso l'alto, la punta verso il basso servira' per l'operazione

inversa. Quando viene utilizzato questo gadget, il gioco in corso viene temporaneamente sospeso. Quest'opzione è certamente utile dal momento che permette al giocatore in caso di distrazione di poter rivedere le mosse effettuate dalla macchina. Quest'opzione come le seguenti è disponibile in qualsiasi momento: opzioni di inizio e fine gioco (Start/Stop Game), opzioni per la scelta dei contendenti (Amiga/Persona, Amiga/Amiga, Persona/Persona), opzione orologio (Clocks), visualizzazione delle mosse permesse (Dispay Legal Moves), visualizzazione di come viene attaccata o difesa una casella (Show How a Square is Attacked/Defended), ecc.

'Chess Analysis', vi permette di risolvere problemi di scacchi, iniziare il gioco da una posizione particolare, ricercare la mossa migliore in una determinata posizione. Il programma vi fornisce le attrezzature necessarie per risolvere i vostri problemi, stara' a voi individuare di volta in volta la più adatta. Il programma offre un'ampia scelta di variabili con cui visualizzare in modo diverso la scacchiera e i pezzi. All'interno della voce 'Special' troveremo un'opzione denominata CHANGE BOARD. Se viene scelta, visualizzera' tre miniature della scacchiera in varie posizioni, a questo punto basterà scecliere quella che ci è più consona. Sempre all'interno di 'Special'troveremo l'opzione CHANGE COLOUR, che ci permette una triplice scelta: REGULAR, CLUB e CAFÈ.

Il programma contiene un modello della scacchiera in tre dimensioni e può esere visto da qualsiasi angolazione lo si desideri. Spostare la scacchiera è come spostare una finestra nell'Amiga. Per vederla da una posizione diversa, ci si posizioni con la freccia su un angolo della scacchiera e si mantenga premuto il pulsante di sinistra del mouse, si sposti a piacere la scacchiera e quando ci si trova nella posizione desiderata si rilasci il pulsante; a questo punto verrà visualizzata la scacchiera sotto il nuovo punto di vista.

Per ridefinire i colori del display ci si posizioni su 'Custom' e si scelga l'opzione CHANGE COLOUR, appariranno sulla parte superiore del video tre controlli che permettono il variare dell'intensita' del rosso, verde e blu sullo schermo.

È possibile ridisegnare i pezzi degli scacchi, utilizzando il Deluxe Paint della Electronic Arts. Dopo aver ridefinito le forme dei vari pezzi, si dovranno registrare sul 'Custom - Pieces Drawer' sul disco contenente il programma sotto i nomi: Pawn, Knight, Rook, Bishop, Queen o King (pedone, cavallo, torre, alfiere, regina, re). Una volta ridisegnato, uno o piu' pezzi, si dovrà eseguire una semplice procedura per installare le nuove figure:

1. Utilizzare il disco Workbenk per far partire l'Amiga nel modo normale.

2. Esegure il Mount del disco contenente il programma di scacchi in DFO:

3. Premere due volte il pulsante sull'icona di 'Art of Chess' per aprire la finestra del programma.

4. Si digiti ora il comando INSTALL.

Verra' visualizzata una nuova

window che fornirà le informazioni necessarie sui progressi della procedura di installazione. Quando l'operazione è terminata, la window sparirà. Ora i pezzi da voi ridisegnati saranno installati. La prossima volta che selezionerete 'Use Custom Pieces' dalla voce 'Project' del menu' i vostri pezzi sostuitiranno i pezzi originali. Per ristabilire le forme precedenti, basterà selezionare nuovamente l'opzione. L'opzione di inizio del gioco con l'Amiga in bianco o in nero, permette l'avvio del gioco nella posizione standard che è quella con tutti i pezzi disposti sulla scacchiera nella posizione di partenza. Comunque, se avete scelto una disposizione differente, il gioco inizierà dalla posizione da voi scelta. Ricordatevi che la macchina non accetta configurazioni nelle quali non siano presenti almeno i due re.

'Quit Current Game' verrà utilizzato per rinunciare ad un gioco, se viene selezionato una seconda volta, tutti i pezzi verranno rimossi dalla scacchiera.

'Load New Game' vi permetterà di caricare in memoria una partita da voi precedentemente registrata o di recuperare una delle partite tra scacchisti famosi che sono disponibili nel disco. In entrambi i casi vi verrà richiesto di digitare il nome della partita che intendete caricare.

'Save Current Game' vi permette di registrare su disco la partita da voi giocata, vi verrà naturalmente richiesto di digitare un nome per identificare la partita che intendete registrare.

'Quit Chess' vi fornira' le istruzioni per terminare il gioco: dovrete togliere il disco contenente il programma e premere contemporaneamente i tasti Control, Amiga e Commodore.

L'opzione 'Show/Clear Board Coordinates' vi permettera' di visualizzare o meno le coordinate delle caselle della scacchiera. 'Show/Hide Title Bar' permette di visualizzare o meno la linea di commento sulla parte superiore dello schermo; quest'opzione vi sara' utile

per visualizzare alcune posizioni dei pezzi ai bordi della scacchiera.

'Print a Game', stampa tutte le mosse del gioco in memoria; che possono essere:

1. Un gioco non terminato.
2. Un gioco appena finito.
3. Un gioco caricato da disco.

Nei tornei le partite vengono giocate a 'tempo', cioe' si debbono fare un determinato numero di mosse entro un lasso di tempo prefissato. L'opzione 'Tournament Rules' vi permette di scegliere una delle seguenti sezioni:

A. 40 mosse per le prime 2 ore e mezzo e 16 mosse per ogni ora seguente (la partita dura 5 ore).

B. 45 mosse per le prime 2 ore e 18 mosse per ogni ora seguente (la partita dura 4 ore).

C. 36 mosse per le prime 2 ore e 18 mosse per ogni ora seguente (la partita dura 4 ore).

D. 40 mosse per le prime 2 ore e 20 mosse per ogni ora seguente (la partita dura 4 ore).

E. 50 mosse per le prime 2 ore e mezzo e 20 mosse per ogni ora seguente (la partita dura 5 ore).

L'opzione 'Play' del menu' principale permette di accedere ad un sub-menù con il quale si può scegliere di cambiare parte in qualsiasi momento del gioco. Si può scegliere di giocare tra due persone, avvalendosi delle facilitazioni permesse dal programma: controllo delle mosse legali, visione delle mosse permesse, analisi delle mosse di difesa e di attacco. Si può scegliere di giocare contro il computer o far giocare il computer contro se stesso; quest'ultima opzione è molto utile in quanto permette alla macchina la risoluzione di problemi nei quali l'utente si è trovato in stato di stallo. Ci si può permettere delle pause durante il gioco scegliendo l'opzione apposita. Si può indurre la macchina a muovere mentre sta valutando la mossa da fare o indicare se è il bianco o il nero che deve iniziare durante la risoluzione di un problema proposto alla macchina.

L'opzione 'Edit'del menu' principale permette il passaggio ad un sub-menù con il quale si può scegliere di posizionare i pezzi sulla scacchiera per ulteriori analisi; si possono disporre i pezzi sulla scacchiera vuota, si possono posizionare i pezzi partendo dalla posizione di inizio partita o partire da una posizione qualsiasi di una partita in gioco.

Del menù principale fa parte anche l'opzione 'Special'che permette di accedere a 'Change Colours' il quale ci mette a disposizione ulteriori quattro sub-opzioni. La prima, detta 'Regular', permette di ristabilire i colori di default. 'Cafe and Club' sono due set di colori diversi. 'Custom' vi permette di adattare a piacimento i colori sulla tastiera. Da questa opzione si puo' selezionare 'Change Board', che fornisce la scelta tra tre scacchiere viste da posizioni diverse. Comunque e' possibile variare l'angolazione della scacchiera senza dover selezionare questo menù, basterà semplicemente posizionare la freccia su un angolo della scacchiera e spostarla mantenendo premuto il pulsante di sinistra del mouse. È possibile variare il tempo di risposta della macchina o il livello di abilità tramite l'opzione 'Change Skill'.

'Show Legal Moves' visualizza le possibili mosse di un pezzo (si prema il pulsante di sinistra del mouse) cambiando il colore delle caselle a cui puo' accedere. Dopo aver scelto 'Show Defence', ci si posizioni sulla casella che si intende difendere si prema il pulsante di sinistra del mouse e in questo modo le caselle contenenti dei pezzi che possono difendere quella posizione cambiano colore. 'Show Attack' opera allo stesso modo dell'opzione precedente visualizzando pero' i pezzi che possono nuocere alla casella indicata. 'Watch the Amiga Think' visualizza sullo schermo l'analisi che effettua il computer prima di eseguire una mossa; la freccia indica quale mossa è considerata la più valida, qual'è la risposta migliore a quella mossa, qual'è la risposta migliore alla risposta, ecc. 'Use Custom/Regular Pieces' permette l'utilizzo dei pezzi definiti dall'utente o con un'ulteriore scelta dal computer.

Scegliendo 'Clocks' si accede ad una serie di scelte: CLOCK ON, CLOCK OFF, ZERO CLOCKS (bianco, nero o entrambi),PAUSE. L'orologio ci indica il tempo utilizzato da un giocatore per effettuare una mossa.

'Sound'ci rimanda ad una duplice scelta: 'Voice Commentary', fornisce un commento parlato del gioco e l'uso di varie facilitazioni; 'Voice Cuing' avvisa semplicemente quale dei due giocatori deve muovere.

Il programma e' provvisto di una libreria nella quale sono registrate 30 partite tra campioni di ogni tempo. Da Wilhelm Steinitz contro Kurt von Bardelben giocata a Mosca nel 1895 a Victor Korchnoi contro Anatoly Karpov giocata a Bagvio City nel 1979.

Il programma "The Art of Chess" è prodotto e distribuito in Italia dalla C.T.O. srl via Indipendenza 40 Bologna.

A 8600: L'

Finalmente un

di Roberto Beccia

Arriva dall'Inghilterra, un po' in ritardo per la verità, un Genlock di progettazione inglese in grado di trattare segnali televisivi di casa nostra, cioè in standard PAL. Compatto e affidabile, capace di produrre immagini di buona qualità, si presenta con le carte in regola per soddisfare le esigenze professionali.

EUROPA VIDEOUNITA

Genlock tutto europeo in standard PAL

Questo Genlock, o meglio "PAL video controller" come amano definirlo i signori della Interactive che lo hanno progettato e costruito verso la fine del 1986, è finalmente distribuito anche in Italia. È frutto di una collaborazione in quanto l'hardware è stato realizzato dall'Interactive, mentre il software fornito a corredo è stato partorito dalla Ariadne Software di Londra. Se questi nomi non vi suggeriscono nulla non vi preoccupate; anche noi ci troviamo nella medesima situazione.

Che cos'è un Genlock?

Certo, se ne fa un gran parlare però, quanti lettori sanno come funziona e a che cosa serve esattamente? Cerchiamo allora di diradare un po' la nebbia che avvolge questo oggetto misterioso.

Genlock non è il nome di un robot uscito da un cartone animato giapponese, bensì un termine che definisce la tecnica usata per sincronizzare due diverse sorgenti di segnale video allo scopo di combinarle assieme come fossero un'unica immagine.

Nel caso particolare che stiamo esaminando, una delle due sorgenti è rappresentata dal segnale proveniente dal computer, mentre l'altra può essere l'uscita di una telecamera, di un videoregistratore o di un lettore di videodischi.

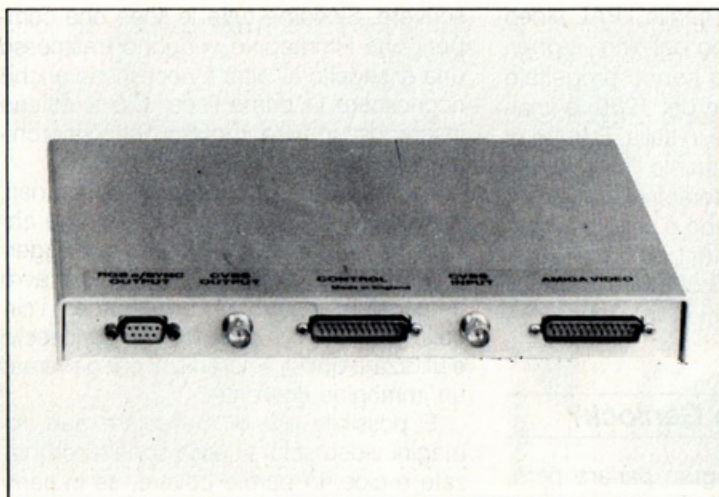
È realmente necessario questo accessorio?

È utile a questo punto fare una breve sosta, per chiarire ai lettori che hanno poca dimestichezza con i segnali video come sia costituita l'immagine televisiva, oppure quella generata dal computer. Entrambe usano una tecnica chiamata *raster* che sta a significare che l'immagine è costituita da un insieme di linee. L'inizio di ogni linea è riconoscibile da un particolare impulso detto sincronismo di riga. Il suo uso è analogo al ritorno carrello nella macchina per scrivere. Siccome tutte le linee che compongono l'immagine vengono trasmesse una di seguito all'altra è necessario anche riconoscere la prima linea. Ciò si ottiene inserendo un altro impulso detto sincronismo di quadro.

Se si tenta di mescolare due segnali, provenienti da due fonti diverse, che abbiano degli impulsi di sincronismo leggermente diversi o sfasati tra di loro, si avrà un'altissima probabilità di confondere i circuiti che hanno il compito di riconoscere e utilizzare questi sincronismi per generare un immagine coerente.

È possibile quindi mescolare due immagini video solo se esse sono sincronizzate e cioè, in parole povere, se iniziano allo stesso istante e viaggiano a braccetto. Ebbene, questo si può fare tranquillamente e dobbiamo ringraziare i progettisti di Amiga per la loro lungimiranza, in quanto hanno previsto la possibilità di sostituire l'oscillatore principale della macchina (generatore di clock) con un segnale proveniente dall'esterno. In questo modo, riferendoci al nostro caso specifico, in presenza di un segnale video in ingresso al Genlock il computer viene asservito ad esso. Quando questo viene rimosso, l'Amiga riprende la sua sincronizzazione interna.

Assodato il fatto che le due immagini si snodano nel tempo in parallelo, rimane da chiarire come è possibile fonderle assieme per ottenerne una sola. Concettualmente è più semplice di quanto si possa immaginare anche se le implicazioni tecnologiche sono abbastanza pesanti. In pratica succede questo: il colore dato dal registro zero dell'Amiga (tutti sanno che esistono 32 registri che contengono le informazioni di colore) viene sostituito dal segnale video proveniente dalla fonte esterna. Ciò è possibile per due motivi ben precisi:

— sappiamo esattamente quando viene usato il registro zero poiché l'Amiga, bontà

Il kit del Genlock al completo.

sua, ci fornisce sul connettore per l'uscita video RGB un segnale denominato ZERO DETECT che evidenzia questo stato;

— questo segnale viene prelevato dal Genlock e usato da un circuito integrato molto complesso (TDA3560, PAL decoder) per effettuare una commutazione fra l'informazione video proveniente dal computer e quella proveniente da una fonte esterna.

Si tenga però presente che questa commutazione avviene ad una velocità elevatissima; infatti deve agire a livello di pixel.

Le applicazioni di questa tecnica sono molteplici e limitate solamente dalla fantasia. Il suo uso in ambiente televisivo è estremamente frequente, basti pensare alle immagini che vengono sovraimpresse a fianco dei giornalisti che commentano i nostri telegiornali. In ambiente casalingo l'applicazione principale riteniamo sia legata alla produzione di titoli e presentazioni grafiche che accompagnino le videoregistrazioni hobbistiche, oramai abbastanza diffuse.

L'A8600 non si limita solamente a sovrapporre due immagini, ma è in grado di fare alcune altre cosette simpatiche che spiegheremo più avanti quando affronteremo l'argomento software.

Che cosa si acquista?

Innanzitutto l'A8600 vero e proprio, contenuto in una scatola metallica. Vengono poi forniti a corredo:

— un trasformatore di alimentazione esterno al Genlock con cavo di uscita terminante in un connettore del tipo "punto-linea" (usati in passato per collegare casse acustiche negli impianti hi-fi di basso costo);

— un cavo, o meglio un moncherino di cavo, visto che è lungo circa dieci centimetri, per la connessione all'uscita RGB del computer;

— un cavo per il collegamento alla porta parallela, un po' più lungo del precedente (circa 25 cm.) e provvisto ad una estremità di due connettori piatti a 25 piedini, l'uno maschio e l'altro femmina, allo scopo di garantire la compatibilità d'uso tanto con gli Amiga 1000 quanto con i 2000 e i 500;

— un cavo per collegare il monitor 1081 all'uscita RGB del Genlock e all'uscita audio del computer; l'A8600 infatti non gestisce la parte sonora e questo compito viene demandato a delle unità esterne;

— un dischetto contenente il software di gestione dell'apparecchiatura;

— un manualino di 18 pagine, essenziale ma esauriente.

Il contenitore, metallico di color avorio, è alto circa tre centimetri e reca sul lato anteriore cinque connettori, il significato dei quali è esplicitato dalle scritte adesive applicate sul coperchio, in corrispondenza dei rispettivi connettori. Vediamo in dettaglio questo aspetto. Da sinistra verso destra troviamo:

— "RGBa/SYNC OUTPUT", connettore piatto a nove pin per il collegamento al monitor 1081;

— "CVBS OUTPUT", uscita del segnale video composito su connettore tipo BNC;

— "CONTROL", connettore piatto a 25 pin per il collegamento alla porta parallela;

— "CVBS INPUT",ingresso del segnale video composito proveniente da una sorgente esterna, su connettore BNC;

— "AMIGA VIDEO", piatto a 25 piedini (la denominazione è autoesplicativa).

Dal lato posteriore esce uno spezzone di cavo terminante con un connettore "punto-linea" per il collegamento al trasformatore di alimentazione.

Uno sguardo all'interno

Aprire il contenitore è un'operazione semplicissima dato che è sufficiente togliere quattro viti poste sui due lati corti della scatola. Togliamo il coperchio con la curiosità tipica dei bambini quando devono aprire un pacco contenente un regalo e, come i bambini che lasciando galoppare la fantasia a briglia sciolta si aspettano di trovare chissà quali cose meravigliose, rimaniamo un po' delusi, non tanto per il contenuto quanto per come questo è stato realizzato. L'ingegnerizzazione non è certo l'aspetto migliore di questo prodotto, anzi la parte più curata, e cioè il contenitore metallico, è forse la meno importante.

La parte principale è costituita da una scheda delle dimensioni approssimative di 24 x 11 centimetri, che occupa circa due terzi del contenitore. Su uno dei lati maggiori sono saldati i tre connettori piatti già visti precedentemente. A loro volta, questi connettori sono avvitati al contenitore, contribuendo così al fissaggio della scheda stessa. Le due prese BNC non costituiscono corpo unico con il circuito stampato, ma sono unite ad esso tramite quattro fili intrecciati a coppie che terminano in un connettorino sfilabile dal circuito stampato stesso.

A questo punto facciamo appello alla benevolenza dei lettori, poiché la descrizione si svilupperà su un piano decisamente più tecnico e quindi meno gradito a coloro che non si occupano direttamente di elettronica. Cercheremo, come affermano varie personalità pubbliche per ingannare le platee, di essere brevi.

Ritorniamo al nostro oggetto. Sul lato opposto rispetto a quello su cui si trovano i connettori è fissata un'altra scheda, molto più piccola della precedente e dall'aspetto più artigianale. Su di essa trovano posto due circuiti integrati regolatori di tensione (7805 e 7812), che hanno il compito di fornire le due tensioni necessarie al funzionamento del genlock (5v e 12v). Ovviamente è presente anche il ponte raddrizzatore ed un adeguato condensatore di filtro. Sempre su questa scheda si trovano anche: un relè, attivato da una tensione di 12v proveniente dal computer, che permette di asservire l'accensione del Gen-

lock a quella dell'Amiga, ed un altro circuito, provvisto di una regolazione, i cui compiti ci sono sconosciuti.

Ad un primo esame risulta evidente che questa scheda è stata aggiunta con l'intenzione di incorporare delle funzioni non previste dalla prima stesura del progetto. Lo denuncia il fatto che i collegamenti con la scheda principale, eseguiti con del filo intrecciato, sono portati in punti non previsti. Altresì evidente risulta l'intenzione iniziale dei progettisti di sfruttare le tensioni disponibili sulla porta RGB dell'Amiga, abbandonata successivamente, probabilmente a causa di un eccessivo consumo dell'apparecchiatura.

Passiamo ora a descrivere la scheda principale. La prima impressione che se ne ricava è che traspare una sorta di schizofrenia nel comportamento dell'autore del circuito stampato. La parte destra, infatti, appare più ordinata della sinistra che, viceversa, ha una densità di componenti degna di una radiolina giapponese. Un altro particolare strano riguarda il percorso delle tre componenti cromatiche (segnale RGB). Le piste partono infatti dal connettore "AMIGA VIDEO" (posto in basso a destra guardando la scheda di fronte) e si snodano lungo più di due lati del circuito stampato per raggiungere la loro destinazione, compiendo in totale un percorso di ben 42 centimetri. Non vi sembra esagerato?

Un'altra sorpresa riservataci dai signori dell'Interactive riguarda i circuiti integrati: erano tutti verniciati in nero allo scopo di occultare le sigle identificative. Abbiamo detto "erano verniciati" perché abbiamo subito provveduto a rimuovere la vernice con un coltellino e con un po' di pazienza. Purtroppo l'impresa è riuscita a metà perché non tutti si sono rivelati. Comunque si è riusciti a capire che:

— la parte di controllo ed interfaccia con l'Amiga è stata realizzata con dei normali TTL della serie 74LS;

— la parte relativa alla decodifica del segnale videocomposito in ingresso e alla commutazione fra questo e il segnale RGB proveniente dal computer, allo scopo di produrre un'unica uscita RGB, è affidata a un circuito integrato della Philips: il TDA3560 (i più curiosi vadano a sbirciare nel manuale della Philips relativo ai circuiti integrati per impiego televisivo e vi troveranno anche il circuito applicativo che, grossomodo, è quasi uguale a quello usato nell'A8600);

— la parte che si incarica di riprodurre un segnale video composito, partendo da quello RGB che si ottiene in uscita al TDA3560, utilizza molto probabilmente (non c'è una sigla a confermare questa ipotesi purtroppo) l'MC1377 della Motorola, che è lo stesso integrato usato nell'Amiga 1000 per generare l'uscita composita.

È necessario rilevare che sulla scheda sono presenti varie regolazioni. Sconsigliamo però l'utente di effettuare dei ritocchi su tutto ciò che si può girare con un cacciavite, a meno che non sia in possesso di un'adeguata strumentazione e conosca bene la materia. Gli unici trimmer che si possono smanettare, anzi a volte è prorio necessario intervenire su di essi, riguardano le regolazioni di ampiezza, saturazione e luminosità del segnale video composito in ingresso. Sono necessarie per bilanciare fra di loro le due sorgenti video in modo da ottenere un risultato armonioso. È molto facile individuarle: si trovano proprio al centro della scheda e sono denominate Amp, Sat e DC Lev.

Uso dell'A8600 e software

Quando si hanno tutti i pezzi a portata di mano si possono effettuare le interconnessioni previste, a computer spento ovviamente. Ci si accorge a questo punto, data la conformazione dei cavi di collegamento, che l'unica collocazione possibile è sotto il contenitore di un Amiga 1000. In questa sede il contenitore dell'A8600 si adatta perfettamente; l'unico inconveniente è costituito dal cavetto di alimentazione che necessariamente deve uscire dal retro del computer, contrastando così la scorrevolezza del cavo della tastiera.

Terminata l'installazione si può accendere il computer, il Genlock si accenderà automaticamente presentandoci, quando il sistema è pronto (alla richiesta del Workbench), la grafica generata dal computer sovrapposta all'immagine proveniente dall'ingresso video composito. Per funzionare in questo modo l'A8600 non richiede alcun segnale di controllo dalla porta parallela essendo questa la condizione di default imposta dall'hardware all'accensione. Si possono però ottenere altre cose che esamineremo tra breve parlando del software.

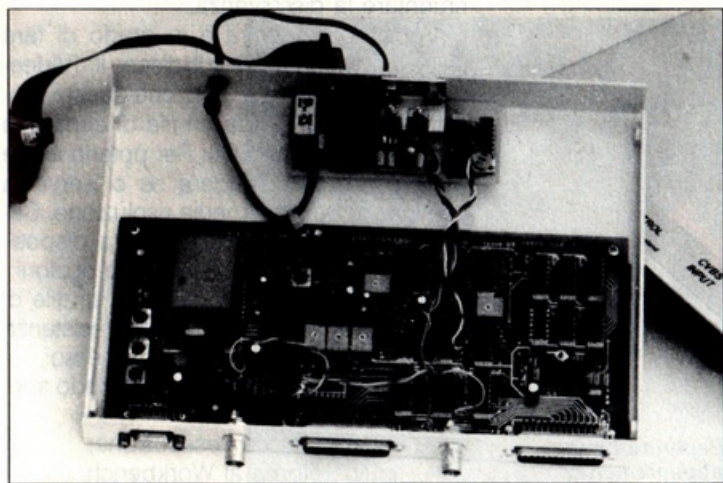
Assieme all'hardware viene fornito un dischetto contenente il seguente software:

— una libreria (genlock.library) che permette il controllo del Genlock da un programma scritto in Basic, C, assembler o qualsiasi altro linguaggio in grado di accedere alle funzioni che la libreria mette a disposizione;

— un "device" (player.device) che permette di comunicare dei semplici comandi, usando la porta seriale, ad un lettore di videodischi esterno (in particolare un LaserVision Philips);

— alcuni file d'interfaccia (genlock.i, genlock.h, genlock.bmap, genlock.lib) che facilitano l'uso della libreria nei programmi applicativi;

— un programma di utilità (Genlock control) che permette di controllare l'A8600 direttamente da Workbench mediante l'uso del mouse;

— un'altra utility (videoshow) che consente il controllo del Genlock per mezzo di un file di comando scritto in codice ASCII, e quindi approntabile e modificabile molto comodamente con un text editor qualsiasi.-

Connettori dell'A 8600.

Per rendere operativo il software è prevista l'installazione dei moduli "genlock.library" e "player.device" (quest'ultimo non indispensabile ovviamente) sul disco del Workbench o su una copia di esso. È chiara l'intenzione di usare due drive, anche se l'utente può ricorrere a vari artifizi per evitarlo. Si installano quindi i due moduli semplicemente cliccando sull'icona "Install Genlock" del disco fornitoci e inserendo il Workbench (se si possiede un unico drive) quando ci viene richiesto. A questo punto ci si può rendere immediatamente conto di cosa sia in grado di fare questo dispositivo semplicemente inserendo il disco con il software di controllo e attivando il tool "Genlock Control", il quale apparirà in una finestra del Workbench. Rimandiamo un attimo la descrizione di questa finestra e soffermiamoci invece sulla programmabilità dell'A8600. Ci sono quattro modi di funzionamento che si possono selezionare attraverso due linee di controllo della porta parallela:

— modo 0 (Colour Transparency Mode), in cui una combinazione selezionata delle tre componenti cromatiche provenienti dall'uscita RGB digitale viene considerata trasparente, permettendo di vedere l'immagine video sottostante (esempio: considerando un range di variazione da 0 a 15 per ognuna delle componenti cromatiche analogiche, se impostiamo il colore ciano, dato dalla somma di verde e blu, tutti i colori che avranno una componente di verde e di blu maggiore o uguale a 8 ed una componente di rosso minore di 8 saranno considerati trasparenti);

— modo 1 (External Picture Only), che visualizza solamente l'immagine esterna;

— modo 2 (Amiga Picture Only), che ci mostra unicamente l'immagine generata dal computer;

— modo 3 (Amiga Overlay Mode), in cui l'immagine esterna si sostituisce al colore di fondo determinato dal contenuto del registro di colore zero, includendo anche il bordo.

È anche possibile programmare la dissolvenza dell'immagine esterna all'apparire (fade in) e allo scomparire (fade out), agendo su un'unica linea di controllo e variando temporalmente la proporzione fra il livello logico alto e quello basso.

Facciamo un passo indietro e ritorniamo alla finestra del tool "Genlock Control". Questa utility ci permette di sperimentare la programmabilità dell'A8600 scegliendo tre dei quattro modi descritti e cioè:

— "Off", che seleziona solo la grafica dell'Amiga;

— "Back", che seleziona il modo di default con il video esterno che sostituisce il colore di fondo;

— "Col", che abilita la trasparenza.

Sempre in questa finestra sono presenti dei gadget per scegliere uno degli otto colori che determinano le combinazioni trasparenti e due aree denominate "up e "down", cliccando sulle quali si può sperimentare la dissolvenza.

Vediamo ora cosa è in grado di fare l'altra utility presente sul disco: il "Videoshow". Abbiamo già detto che questo programma è pilotato da un file di comando scritto in caratteri ASCII. Per poterlo usare dobbiamo però decidere se ci serve la versione in bassa o media risoluzione. Caricato il programma, abbiamo a disposizione un solo menù con quattro opzioni:

— "load", carica in memoria un file di comando specificato e verifica l'esistenza di tutti i files grafici richiamati in esso:

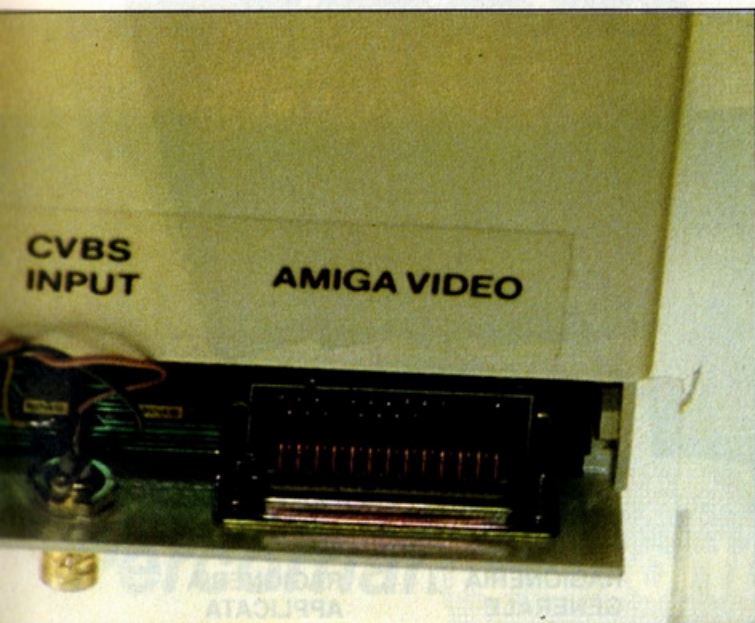
— "run", esegue il file di comando specificato;

— "stop", blocca l'esecuzione;

— "quit", ritorna al Workbench.

Paperino ha un appuntamento.

Non si illudano i lettori di poter raccontare le loro disgrazie nei files di comando (che, per comodità, da ora in poi chiameremo script, adeguandoci alla terminologia usata nel manuale), e pretendere che Videoshow sia capace di comprensione. Questo tool è un pochino più limitato, anzi capisce solamente questi comandi:

end .... termina l'escuzione dello script
repeat .... riprende dall'inizio lo script
delay <tempo> .. effettua una attesa determinata da <tempo>
mouse .... attende un comando dato con il mouse
exec <comando> ..... invoca un altro programma esterno a Videoshow
slide <nome> .... visualizza lo schermo <nome>
overlay <x> <y> <nome> .... pone sullo schermo la finestra <nome> alle coordinate <x> e <y>
fadein <tempo> <nome> .. dissolvenza positiva per lo schermo <nome>, della durata di <tempo>
fadeout <tempo> .... dissolvenza negativa per lo schermo corrente, della durata di <tempo>
colour <reg> <valore> ... imposta un colore determinato da <valore> nel registro di colore <reg>
fadecolour <reg> <val> <t> ... porta il registro <reg> al colore <val> nel tempo <t>
videomode <modo> ...... seleziona il modo di funzionamento
videocolour <col> .. <col> definisce la trasparenza
videofadeout <t> .. scurisce l'immagine video fino al nero nel tempo x t>
videofadein <t> .... l'inverso
sendplayer <stringa> ...... invia il comando <stringa> al lettore di videodisco
waitplayer <stringa> ...... invia il comando <stringa> e attende l'esecuzione

Le porte esterne del genlock.

Concludiamo accennando al fatto che è possibile controllare l'A8600 anche da un programma scritto dall'utente. Le funzioni messe a disposizione sono in questo caso ristrette alle operazioni base effettuabili agendo sulla porta di controllo. Gli effetti più complessi sono lasciati alla buona volontà del programmatore.

Conclusioni

Tirando le somme, questo benedetto A8600 è una macchina valida? Certo non fa grandi cose, considerato anche che il prezzo di vendita non mette allegria, però, a nostro giudizio, svolge onestamente il lavoro per cui è stato progettato. In mano ad un programmatore smaliziato si possono sicuramente ottenere dei risultati degni di rilievo. Ribadiamo la nostra impressione secondo la quale questo oggetto si rivolge ad un mercato professionale perché è privilegiata come qualità l'uscita RGB rispetto a quella per videoregistratore.

Confrontate le due sul nostro fedele monitor 1081, la differenza salta agli occhi: l'immagine RGB è pulita mentre quella relativa al videocomposito e decisamente scadente. Al pubblico l'ardua sentenza!

Sotto le armi.

Il Genlock A8600 ci è stato gentilmente prestato dalla "Informatica Italia", Corso Re Umberto 128 Torino tel. 501647, che ne gestisce la distribuzione.

Percorriamo i tortuosi sentieri del CLI

di Alessandro Prandi

Come d'obbligo, per l'inizio di un lungo cammino da intraprendere assieme, abbiamo scelto di trattare il DOS Amiga sin da questo numero.

Tralasceremo il solito elenco di comandi DOS, oramai trattati esaurientemente da testi e riviste, per passare invece ad un esame più dettagliato delle varie parti del sistema operativo di questa meravigliosa macchina. L'intinerario che ci siamo prefissi di seguire questo mese, ci porterà all'interno dei file comando, per poi continuare verso lo screen editor del DOS e, infine, portarci all'esplorazione del disco Workbench.

Apriamo le finestre

Aprendo una finestra CLI, in pratica, ci si apre una strada nel disco del Workbench, infatti da questa finestra possiamo gestire delle operazioni che il Workbench non ci consente, come ad esempio creare una Startup-Sequence, accedere alla RAM Disk o riassegnare le device logiche. Il CLI, di per sè, è un qualcosa di piuttosto primitivo, generalmente una finestra CLI esegue solo i nomi dei file digitati dall'utente. Fortunatamente il DOS Amiga dispone di modi di gestione del CLI che ci permettono di aggirare certi ostacoli. Uno di questi consiste nel creare dei file comando, altrimenti chiamati file batch o exec. Essi contengono semplicemente una sequenza di comandi da far eseguire al DOS dell'Amiga. Per richiamare un file comando scrivete l'istruzione EXECUTE seguita dal nome del file. A differenza dell'MS-DOS, l'Amiga-DOS non riconosce automaticamente un file comando, pertanto dovrete sempre specificare l'istruzione EXECUTE. In questo modo verrà prima esaminata la directory corrente per trovare il file, e se la ricerca non ha successo allora verrà esaminata la device S che normalmente è la directory s del disco da cui siete partiti. La device logica S: può contenere uno speciale file comando chiamato Startup-Sequence, il quale viene eseguito automati-

camente inserendo il disco Workbench dopo aver acceso il computer. Ecco una possibile Startup-Sequence:

```
Echo "Workbench disk. Release 1.2"
Echo ""
Date < S:ora
Echo "Inserire la data e l'ora:"
Date > Nil: ?
Date > S:ora
Echo ""
Load WB
EndCLI > Nil:
```

```
BindDrivers
echo "A2000 Workbench 1.2 V33.56 22-MAY-87*N"
Assign WB: sys:
IF EXISTS sys:s/JH0
   echo "Initializing Janus ...*N"
   Wait 25
   Echo "Mounting Janus Hard Disk*N"
   DJMount
   IF EXISTS JH0:
      Assign WB: JH0:
      Echo "Transfering control to JH0: *N"
   ELSE
      Assign WB: sys:
   EndIF
EndIF
IF EXISTS sys:s/DH0
   Assign WB: DH0:
   Echo "Transfering control to DH0: *N"
EndIF
AddBuffers WB: 30
CD WB:
IF EXISTS WB:c
   Assign c: WB:c
EndIF
IF EXISTS WB:s
   Assign s: WB:s
EndIF
IF EXISTS WB:t
```

Schermata dell'Editor video di Amiga.

Questa sequenza di comandi rende più semplice l'aggiornamento dell'orologio del sistema nel caso non disponiate della versione Amiga con batteria. La terza riga legge il file S:now per vedere l'ultima data. La quinta riga, Date ×Nil: ?, vi chiede la nuova data, mentre la sesta deposita la data nel file S:now. Dopo aver settato la data il file comando carica il Workbench e chiude la finestra CLI. Il comando Echo semplicemente stampa le specifiche stringhe sullo schermo. I file comando sono molto utili per quanto riguarda tutte quelle operazioni che normalmente portano via molto tempo nel scriverle, ad esempio se volete spostare dei file dovete copiarli in una nuova locazione e quindi cancellare gli originali, il che comporta una notevole perdita di tempo. Il seguente file comando lo farà per voi:

```
.Key NomeVecchio,NomeNuovo
;MOVE sposta il file alla nuova locazione e
cancella l'originale FailAt 21
If > nil: Not Exists <NomeVecchio>
  Echo "*" <NomeVecchio> *"non esiste."
  Quit
Else
  If Exists <NomeNuovo>
   Echo "*" <NomeNuovo> *"già esistente."
  Quit
  Else
    Copy <NomeVecchio>  <NomeNuovo>
    Delete <NomeVecchio>
    Endif
Endif
```

La seconda riga è solo un commento per ricordarvi cosa fa il file. La prima riga legge gli argomenti e li chiama NomeVecchio e NomeNuovo. Gli argomenti sono i nomi dei file che avete scritto dopo il comando. Nella terza riga, FailAt ressetta il più basso codice di errore il quale costringe il file comando all'arresto. In questo caso, settando FailAt a 21 gli si impedisce di fermarsi in caso l'utente abbia dimenticato gli argomenti. La riga seguente verifica l'esistenza di NomeVecchio; nel caso che il file non esista, apparirà il messaggio: "NomeVecchio non esiste." e quindi terminerà l'Execute. In tal caso il DOS comunque controlla se esiste il file NomeNuovo; se c'è, il file comando si ferma in modo da non permettervi di scriverci accidentalmente sopra. Infine se tutti i parametri sono esatti il file comando esegue una copia di NomeVecchio, lo salva come NomeNuovo e quindi cancella il primo. Move vi illustra solo alcuni degli aspetti dei file comando avanzati. Provate a fare da voi alcuni semplici file comando e noterete in breve tempo come si possa velocizzare e migliorare tutte le operazioni che normalmente richiedono un lungo e noioso iter.

ED... itor

Per creare questo tipo di file potete richiamare lo Screen Editor, ED. ED è un piccolo e maneggevole editor che voi potete usare anche per la stesura di programmi in C o Basic. Per editare un file dovete scrivere ED seguito dal nome del file e, se lo volete, dalla dimensione del buffer del file, che è la parte di memoria dove il file viene depositato mentre voi ci lavorate.

```
ED <nomefile> (SIZE n)
```

Il buffer è per default di 40 K-byte. Se voi pensate di non riuscire ad editare un programma poiché ritenete di non avere abbastanza memoria, allora diminuite il valore di SIZE. Una volta entrati nell'editor ED potete muovervi con i tasti cursore, inserire del testo e richiamare due tipi di comandi: immediati ed estesi. Per eseguire i comandi in modo immediato dovete premere il tasto control (CTRL) e la lettera desiderata. Per esempio CTRL-A inserisce una riga dopo la riga dove si trova il cursore; CTRL-B cancella la riga corrente; CTRL-Y cancella i caratteri alla destra del cursore; CTRL-D fa scrollare il testo verso il basso; CTRL-U fa scrollare il testo verso l'alto. Per eseguire invece i comandi in modo esteso dovete premere il tasto ESC. A questo punto il computer aspetta un vostro comando e che premiate il RETURN. I comandi estesi comprendono: B, per spostarvi alla fine del file; F /stringa/ per la ricerca di una stringa; J per collegare la riga corrente a quella seguente; Mx n xper spostarsi al numero di righa n; Q per uscire dal file senza salvare il testo; SA per il save del testo; X per salvare il testo e quindi uscire da ED. Per una lista completa dei comandi vedere la tavola 1.

ED non è un vero e proprio wordprocessor ma esso offre alcuni aspetti interessanti per quanto riguarda i file testo, come per esempio i margini e la gestione delle parole a fine riga: ad esempio, se state scrivendo una parola che oltrepassa il margine destro, ED la riporta interamente a capo. I margini sinistro e destro si possono settare con i comandi estesi SL e SR. I comandi BS e BE selezionano l'inizio e la fine di un blocco di testo che voi potete cancellare, copiare e scrivere in un file con i comandi DB, IB e WB respettivamente. Ovviamente ED non può dare una forma al testo o fornire i vari tipi di caratteri (fon-

ts), però, sicuramente, è molto compatto ed estremamente facile da usare.

Altre directory

L'ED e il resto dei comandi DOS dell'Amiga risiedono nel device logico C:, il quale assume la directory c del disco Workbench. Se scrivete il comando ASSIGN senza parametri il DOS vi comunica quale dispositivo logico, i dispositivi di Input/Output ed i volumi dei dischi in quel momento definiti. Il comando ASSIGN può aggiungere, cambiare o cancellare i dispositivi logici predisposti. Ora diamo una breve occhiata ad alcuni device del DOS. I dispositivi logici assumono le directory aventi lo stesso nome sul disco iniziale.

SYS: rappresenta il disco sistema, ovvero quando si inizializza il sistema, SYS: disegna la directory principale del disco di partenza. Il Workbench cerca in SYS: la directory del Sistema, la quale contiene i tool di DiskCopy e Initialize. Il dispositivo C: contiene i comandi CLI, che voi potete rinominare a vostro piacimento, infatti potete inserire qualsiasi programma nel C: e il DOS dell'Amiga lo tratterà come un comando CLI.

Le librerie del Sistema Operativo risiedono in L:. Il Ram-Handler manipola la RAM-disk, ed è caricato in memoria la prima volta che viene copiato un file nella RAM:. Guardandola da CLI la RAM-disk appare come un vero e proprio drive a dischi. Oltre al Ram-Handler, L: contiene il file Disk-Validator che vi permette di verificare l'integrità del disco inserito.

La directory S: o Sequenza, contiene dei file comando. L'AmigaDOS cerca il dispositivo LIBS: per trovare le librerie del sistema. L'Amiga raccoglie le routine connesse tra loro in librerie. Alcune librerie vengono depositate tramite Kickstart nella memoria protetta da scrittura e vi rimangono finché non spegnete la macchina. Altre librerie sono caricate dalla directory LIBS: nella memoria di lettura/scrittura come desiderato; esempi ne sono la info.library, la quale maneggia il comando info del Workbench, e la translator.library la quale traduce il testo in fonemi per il sintetizzatore vocale. Quando le librerie del disco base non sono adoperate, rimangono nella RAM finché un programma non necessita dello spazio di memoria da esse occupato.

Il dispositivo di Input/Output dei drive risiede nel device DEVS:. L'AmigaDOS carica tutto in RAM quando inviate dei dati al modem, alla stampante o al sintetizzatore vocale. Il parallel.device e il serial.device gestiscono le relative porte, parallela e seriale. Il narrator.device sintetizza la voce, il printer.device principalmente filtra i dati da inviare alla stampante da voi selezionata nel Preference ed invia quindi i dati alla porta seriale o parallela. Le definizioni di ciascuna stampante risiedono nella subdirectory printers di DEVS:.

FONTS: immagazzina le varie opzioni per le serie di caratteri. Il Macintosh usa i nomi delle città per catalogare le varie serie di caratteri, l'Amiga invece usa i nomi di alcune pietre preziose. I vari stili di scrittura sono visibili nel tool Notepad nel drawer delle Utilities nel disco del Workbench. Lo stile Romano usato dal Workbench e dal CLI non si trova in FONTS: ma nella ROM del kernel contenuta nello Kickstart. Come le device delle librerie e dell'I/O le varie serie di caratteri vengono caricate in memoria solo all'occorrenza.

L'uso di ASSIGN

Ora rivediamo un attimo il comando ASSIGN, esso dopo aver visulizzato la lista delle assegnazioni ai device logici, mostra i dispositivi fisicamente utilizzabili che possono essere DF0, drive interno; DF1, secondo drive; DH0, drive hard-disk; PAR e

Requester line dell'Editor.

Sommario dei comandi ED

COMANDI IMMEDIATI

Movimento Cursore:

CTRL-D	Scrolla verso il basso mezza schermata
CTRL-U	Scrolla verso l'alto mezza schermata
CTRL-E	Porta il cursore in alto nello schermo o in fondo nello schermo a seconda di dove esso si trovi
CTRL-I TAB	Sposta il cursore alla prossima posizione del tabulatore
CTRL-R	Sposta il cursore alla fine della parola precedente
CTRL-T	Sposta il cursore all'inizio della prossima parola
CTRL-]	Sposta il cursore alla fine della riga, o all'inizio se esso si trova già alla fine

Inserzione/Cancellazione Testo:

CTRL-A	Inserisce una riga dopo la riga corrente
CTRL-B	Cancella la riga corrente
CTRL-H	Cancella il carattere a sinistra del cursore

BACKSPACE
CTRL-O Cancella la parola o lo spazio successivi
CTRL-Y Cancella dal cursore alla fine della riga

Miscellanea:

CTRL-F Inverte il maiuscolo/minuscolo e si sposta a destra
CTRL-G Ripete l'ultimo comando esteso
CTRL-M Carrello di ritorno
RETURN
CTRL-V Ridisegna lo schermo
CTRL-[Per entrare nel modo 'comando esteso'
ESC

COMANDI ESTESI
Iniziano con ESC e finiscono con RETURN

Movimenti Cursore:

ESC-B Sposta il cursore alla fine del file
ESC-CE Sposta il cursore alla fine della riga
ESC-CL Sposta il cursore a sinistra senza cancellare
ESC-CR Sposta il cursore a destra
ESC-CS Sposta il cursore all'inizio della riga
ESC-Mn Sposta il cursore alla riga n
ESC-N Sposta il cursore all'inizio della riga seguente
ESC-P Sposta il cursore all'inizio della riga precedente
ESC-T Sposta il cursore all'inizio del file

Inserzione/Cancellazione Testo:

ESC-A/stringa/ Inserisce una riga dopo quella corrente
ESC-D Cancella la riga corrente
ESC-DC Cancella il carattere sotto il cursore
ESC-I/stringa/ Inserisce una riga prima di quella corrente
ESC-U Cancella eventuali modifiche sulla riga corrente

Manipolazione di Blocchi:

ESC-BE Definisce la fine del blocco
ESC-BS Definisce l'inizio del blocco
ESC-DB Cancella il blocco
ESC-SB Mostra il blocco sullo schermo
ESC-WB Scrive il blocco nel file

Ricerca Testo:

ESC-BF/stringa/ Trova la stringa cercando all'indietro
ESC-E/str1/str2/ Da il nome della stringa2 alla stringa1
ESC-EQ/str1/str2/ Come sopra, solo che prima di cambiare chiede Y/N
ESC-F/stringa/ Trova la stringa cercando in avanti
ESC-LC Considera il maiuscolo/minuscolo nella ricerca
ESC-UC Non considera il maiuscolo/minuscolo nella ricerca

Manipolazione dei File:

ESC-IF/nomefile/ Inserisce un file
ESC-Q Esce senza salvare il testo
ESC-SA Salva il testo
ESC-X Salva il testo ed esce

Margini:

ESC-EX Estende il margine destro
ESC-SLn Setta il margine sinistro
ESC-SRn Setta il margine destro
ESC-STn Setta la distanza dei tabulatori

Miscellanea:

ESC-J Unisce la riga corrente con la successiva
ESC-RP Ripete il comando fino all'errore
ESC-S Interrompe la riga all'altezza del cursore
ESC-SH Mostra le informazioni sul testo

SER, le porte parallela e seriale; PRT per la stampante; ed infine CON e RAW per la console del sistema. Voi potete specificare un dispositivo CON o RAW descrivendo la posizione e la misura di una finestra sullo schermo, per esempio:

CON:da sinistra/dall'alto/larghezza/altezza/nome

CON e RAW differiscono in quanto CON disabilita i tasti funzione ed i tasti cursore ed interpreta certi codici di controllo, mentre RAW invia i caratteri così come digitati. L'output sullo schermo da parte di CON e RAW sembra uguale per entrambi, con l'eccezione che CON impedisce una normale gestione dello schermo e il ritorno del carrello obbliga l'apertura di una nuova riga. Un paragone per evidenziare la differenza tra i due modi, può essere fatto tra i vari wordprocessor che usano il modo RAW, e il CLI che usa il modo CON. I comandi ed i programmi ai quali viene dato il run dalla finestra CLI normalmente inviano l'output nella stessa finestra. Voi potete però facilmente reindirizzare l'input e l'output a diversi device e persino a dei file. Il segno di maggiore (>) serve a reindirizzare l'output;il segno di minore quello dell'input. Per esempio, consideriamo per un attimo il comando Date. Scrivendo solo DATE sia l'ora che la data del sistema appaiono nella finestra CLI. La finestra CLI è lo standard del dispositivo di I/O dato dal simbolo "*".

"DATE > ora" mette la data in un file chiamato ora
"DATE > CON:50/50/300/100/Data" scrive la data in una finestra chiamata Data
"DATE ?" vi suggerisce come immettere la data nella finestra CLI
"DATE < Data" preleva la data dal file Data
"DATE < CON:50/50/300/100/Data < NIL: ?" apre una finestra chiamata Data, nella quale potete scrivere la nuova data del sistema

Voi potete reindirizzare l'output di ciascun comando CLI a seconda delle vostre esigenze. Alcuni comandi CLI considerano i device come argomenti. Per esempio, NEWCLI <device> aprirà una finestra CLI nel specificato device, come in:

NEWCLI CON:0/0/450/150/CLI

Sfortunatamente, è alquanto difficile cambiare i valori di default e la posizione della finestra CLI aperta clickando l'icona CLI del workbench.

Il CLI vi presenta un pertugio dal quale passare per entrare nel mondo dell'AmigaDOS. Se siete abituati a lavorare in MS-DOS vi troverete come a casa vostra nell'uso del CLI.

IN TEMPO REALE

Un esempio di come creare e muovere in tempo reale una immagine tridimensionale usando il Blitter dell'Amiga

di Paolo Russo

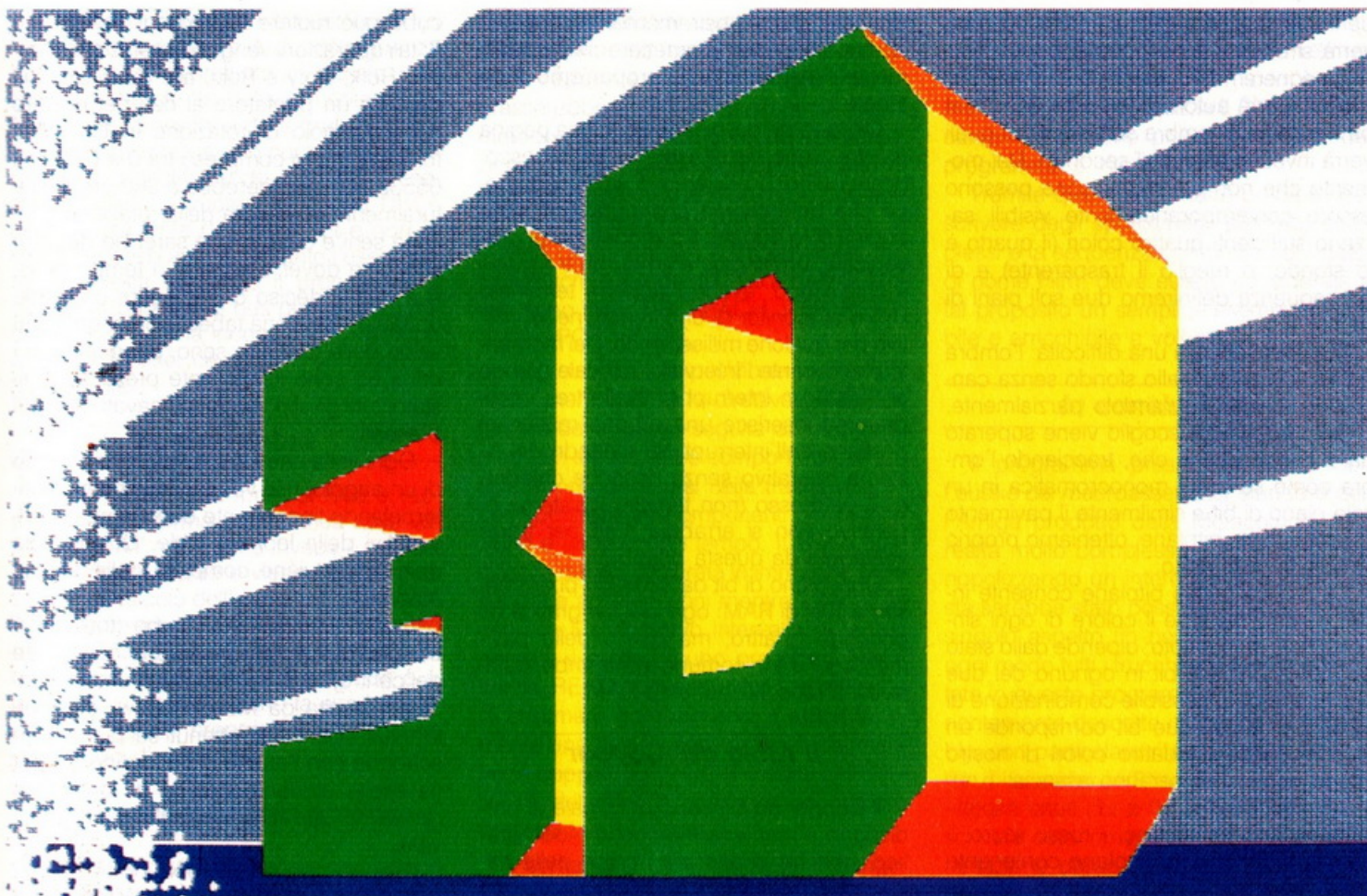
Che l'Amiga possieda una grafica splendida è un fatto risaputo; l'intera macchina è stata primariamente progettata come un computer grafico, e il fatto che non possieda di serie un coprocessore aritmetico, molto diffuso invece nel mondo MS-DOS, bensì un coprocessore grafico dovrebbe essere sufficiente per dimostrarlo a chiunque. Questo è il motivo per cui circolano attualmente miriadi di programmi dimostrativi sulle capacità grafiche dell'Amiga: il giocoliere in ray tracing, il gatto digitalizzato che passeggia tranquillo, per non parlare degli slide show, ormai disponibili a carrettate.

Tutti questi programmi dimostrano più che egregiamente che l'Amiga possiede una risoluzione di 640x400 e che può mostrare 4096 colori contemporaneamente in modo HAM, ma trascurano completamente o quasi un aspetto fondamentale dell'Amiga: la sua velocità grafica. Gli slide show si limitano a caricare passivamente immagini da disco, il giocoliere e il gatto ne differiscono solo per il fatto di mantenere tutte le schermate in RAM; ma il Blitter, capace di tracciare immagini in tempo reale, che fine ha fatto?

Il programma presentato in queste pagine anima un cubo con l'ausilio del tanto bistrattato Blitter, raggiungendo una velocità di tracciamento pari a venticinque immagini al secondo, ossia ciò che viene tecnicamente chiamato 'tempo reale'. La fluidità e varietà di movimento ottenibili grazie a questa tecnica dovrebbero soddisfare chiunque.

Cos'è un'animazione in tempo reale

Una sequenza di immagini molto simili l'una all'altra che si succedono a ritmo elevato creano nel cervello umano l'impressione del movimento. Maggiore è la frequenza alla quale si alternano le immagini, maggiore risulta questa sensazione. Ma cosa si intende per 'tempo reale'? Ormai questa locuzione ha assunto il significato popolare di 'alquanto in fretta', e viene sovente utilizzata perfino quando si parla di certi simulatori di volo che riescono a malapena a tracciare sullo schermo due o tre immagini al secondo.

Tecnicamente si definisce tempo reale la frequenza con la quale le immagini si succedono su di un qualunque monitor o televisore. Negli Stati Uniti vige lo standard NTSC che prevede trenta immagini al secondo (o, più esattamente, sessanta semiquadri), mentre in Europa è comunemente impiegato il PAL che ne impone solo venticinque, in cambio però di una maggiore risoluzione verticale. Una bella differenza

rispetto alla velocità tipica di molte animazioni!

Un semplice cubo

Il programma illustrato in quest'articolo si prefigge un obiettivo semplicissimo: muovere un cubo in tempo reale. Certo, un cubo è davvero una cosa molto banale; di conseguenza, allo scopo di non semplificare la vita al programmatore in misura eccessiva, introdurremo una serie di caratteristiche che renderanno il progetto un tantino più complesso, a tutto vantaggio del realismo dell'animazione.

Innanzitutto sarà richiesto il tempo reale, inteso in senso letterale; il cubo dovrà poi essere visualizzato privo delle linee nascoste e con le facce interamente colorate, senza quindi accontentarsi di tracciarne il perimetro. Inoltre la sfumatura di colore delle varie facce dovrà variare a seconda dell'angolo di illuminazione; introdurremo comunque una certa quota di luce diffusa in modo da rendere visibili anche le facce prive di illuminazione diretta. Il cubo si muoverà su uno sfondo senza cancellarlo: per semplicità ci limiteremo a un cielo azzurro e a una pavimentazione a scacchi bianchi e rossi (non vi ricorda una certa sfera?).

Dulcis in fundo, il cubo proietterà un'ombra semitrasparente sul terreno, che oscurerà il pavimento solo in parte. Non occorre sottolineare la necessità di evitare che l'ombra cancelli il cubo da cui è prodotta a causa di un meccanismo di sovrapposizione non abbastanza sofisticato.

Qualche piccolo trucco

Qualcuno si sarà probabilmente reso conto che l'impresa appena descritta è al di sopra delle possibilità di un normale Amiga (normale perché se possedete una compilation di schede 68020, 68030 e multi-transputer assortite funzionanti in parallelo le cose cambiano), a meno di non fare ricorso a qualche trucco malefico. Uno dei problemi più spinosi è quello delle intersezioni tra le figure bidimensionali del cubo, dell'ombra e dello sfondo; il metodo più efficace per eliminare quello che altrimenti costituirebbe un vero incubo consiste nel risolvere le intersezioni via hardware ricorrendo al modo grafico dual playfield. Al cubo destineremo un intero playfield, mentre l'ombra e il pavimento utilizzeranno due diversi bitplane del secondo playfield. Sarà a questo punto opportuno chiarire l'esatta natura di questo simpatico modo grafico.

Il dual playfield

Solitamente un computer è in grado di visualizzare sullo schermo una sola immagine alla volta; l'Amiga può gestirne due sovrapposte, sia pure con un numero ridotto di colori ciascuna, che prendono per l'occasione il nome di playfield, ossia campi di gioco. La sovrapposizione agisce nel seguente modo: ogni playfield possiede un certo numero di colori numerati da zero in su, ma uno di questi, e precisamente il numero zero, viene chiamato colore di fondo e viene considerato dall'hardware come trasparente. Uno dei due playfield, generalmente il numero uno, viene posto idealmente davanti all'altro e viene quindi visualizzato di preferenza, ma in tutti i suoi punti in cui compare il colore numero zero viene invece mostrato l'altro playfield.

In parole povere, il primo playfield può essere sforacchiato a volontà e attraverso le aperture si vede il secondo. Ogni playfield può inoltre essere scrollato indipen-

dentemente, ma questa caratteristica non verrà sfruttata nel nostro caso.

Disegneremo il cubo nel primo playfield e ciò lo porrà automaticamente davanti al pavimento e all'ombra sottostanti, ai quali verrà invece riservato il secondo. Dal momento che non più di tre facce possono essere contemporaneamente visibili saranno sufficienti quattro colori (il quarto è lo sfondo, o meglio il trasparente) e di conseguenza definiremo due soli piani di bit.

Ma esiste ancora una difficoltà: l'ombra dovrà sovrapporsi allo sfondo senza cancellarlo, bensì oscurandolo parzialmente. Questo apparente scoglio viene superato dalla constatazione che, tracciando l'ombra come se fosse monocromatica in un solo piano di bit e similmente il pavimento in un secondo bitplane, otteniamo proprio il risultato desiderato.

La grafica a due bitplane consente infatti quattro colori, e il colore di ogni singolo pixel, com'è noto, dipende dallo stato dei corrispondenti bit in ognuno dei due bitplane: a ogni possibile combinazione di valori per questi due bit corrisponde un diverso colore. I quattro colori di nostro interesse, ai quali saranno associati i numeri binari 00, 01, 10 e 11, sono rispettivamente il rosso acceso, il rosso scuro, il bianco e il grigio; al bitplane contenente l'ombra è associato il bit di destra, e risulta allora evidente che proprio questo bit decide se il rosso è acceso o scuro e se il bianco è chiaro oppure grigio. Il trucco consiste nella scelta oculata dei numeri da associare ai colori.

Il ruolo del Blitter

Il Blitter possiede molte caratteristiche simpatiche e il presente programma ne sfrutta un paio: la capacità di tracciare linee e di riempire superfici delimitate. Il Blitter viene impiegato una prima volta nella sezione di inizializzazione per disegnare il pavimento a scacchi, che non verrà in seguito più toccato, e lavora quasi ininterrottamente all'interno del loop di esecuzione principale, prima per tracciare i contorni delle facce del cubo e della loro ombra, poi per riempire di colore le aree così delimitate; esso viene anche sfruttato dalla routine di cancellazione delle pagine grafiche. "Perché l'uso del plurale?", si chiederà qualcuno. Ebbene, di pagine grafiche ne esistono ben due, essendo questo un requisito indispensabile per l'impiego della tecnica nota come double buffering. Ci troviamo infatti di fronte a un ulteriore problema: tracciare qualcosa nella pagina grafica correntemente visualizzata può produrre sfarfallii estremamente antiestetici, capaci di compromettere irrimediabilmente il realismo che ci proponiamo di ottenere.

Occorre quindi disegnare in una pagina mentre viene visualizzata l'altra ed escogitare quindi un sistema di sincronizzazione che determini la commutazione della pagina solo durante il cosiddetto intervallo verticale, unico momento durante il quale il chip video, avendo appena terminato l'invio al monitor di una immagine, è inattivo per qualche millisecondo. Dal momento che durante l'intervallo verticale giunge al 68000 un interrupt di livello tre, il programma inserisce una propria routine di gestione dell'interrupt nei meandri del sistema operativo senza neanche chiedergli il permesso (non temete, il sistema operativo non si arrabbia mai) per trarre vantaggio da questa possibilità.

Ogni piano di bit da 320x256 pixel consuma 10K di RAM: ogni pagina grafica ne possiede quattro, ma quello della pavimentazione è in comune a entrambe, quindi occorrono 70K di chip RAM.

Il ruolo del Copper

Il Copper ha lo scopo principale di predisporre e mantenere un certo modo grafico, nonché di alterare i colori della paletta in zone dello schermo ben determinate. Nelle intenzioni di chi ha scritto il sistema operativo il Copper non dovrebbe mai essere pilotato direttamente da un programma, bensì dal sistema operativo stesso; la gestione del Copper che il sistema operativo consente è tuttavia così inefficiente che si è preferito accedervi direttamente.

Grazie al Copper è possibile cambiare a metà schermo il colore di fondo, che passa quindi dall'azzurro del cielo al rosso del terreno, nonché disattivare il secondo playfield nella metà superiore dello schermo allo scopo di limitare il DMA del chip video al minimo indispensabile e aumentare al massimo la velocità del programma (ciò non provoca il minimo problema, risultando alquanto improbabile che il terreno o l'ombra possano trovarsi in cielo).

Solitamente il Copper gestisce anche i puntatori ai bitplane, ma si è preferito in questo caso affidare in parte tale compito al 68000 o, più esattamente, alla routine di gestione dell'interrupt.

Le routine aritmetiche del programma

Le coordinate dei vertici del cubo sono misurate rispetto al centro dello stesso. Il cubo può ruotare intorno agli assi X, Y e Z: tali operazioni vengono gestite dalle routine Rotx, Roty e Rotz, alle quali bisogna passare un puntatore ai dati nel registro A5 e l'angolo di rotazione in D0, sotto forma di intero compreso tra 0 e 65535; il 65536 corrisponderebbe a 360 gradi. Naturalmente la formula della rotazione contiene seni e coseni, che sarebbe davvero seccante dover calcolare in tempo reale; si è quindi deciso di impiegare una look-up table, ossia una tabella contenente 256 valori della funzione seno, dei quali solo i primi 65 sono fisicamente presenti nel listato: tutti gli altri vengono ricavati da quelli presenti.

Ogni volta che occorre calcolare il seno di un angolo, tale valore viene ottenuto interpolando linearmente due elementi consecutivi della look-up table. Quest'ultima operazione viene compiuta dalla routine Alpha.

Dopo aver ruotato il cubo (traslarlo è banale, basta incrementare le coordinate del centro) occorre calcolare l'ombra, funzione assolta da MakeShadow, e proiettare tutti i punti così ottenuti sul piano dello schermo con l'ausilio della routine Project (la scelta alquanto infelice di tale nome deriva da una temporanea crisi di creatività).

A questo punto è opportuno tracciare i confini delle varie aree con Wires che a sua volta si serve di Draw e colorarle richiamando più volte la funzione Fill, che riempie di colore un solo bitplane alla volta. Diverse chiamate alla routine Clear, che cancella un bitplane con l'aiuto del Blitter, sono state sparpagliate tra un'attivazione e l'altra delle routine aritmetiche in modo da parallelizzare per quanto possibile il lavoro del Blitter e quello del processore. A questo proposito sarà opportuno spendere qualche parola sul trucco della divisione a vuoto. Il 68000 e il Blitter si contendono l'uso del bus di sistema e il parallelismo ottenibile è limitato da questo problema.

Quando però il 68000 non ha nient'altro da fare se non attendere che il coprocessore grafico abbia terminato l'operazione affidatagli sorge il problema di impedire che il 68000 occupi inutilmente il bus e ostacoli così l'attività del Blitter. La soluzione escogitata dai progettisti dell'Amiga consiste nel settaggio di un particolare flag, chiamato popolarmente (dalla non folta schiera dei programmatori che conoscono il coprocessore) bit di cattiveria del Blitter; settando questo flag si concede al Blitter la priorità assoluta di accesso al bus, bloccando così il 68000 mentre il coprocessore è attivo. Disgraziatamente questo metodo impedisce al microproces-

sore di rispondere agli interrupt entro un tempo ragionevole ed è di conseguenza sconsigliabile, soprattutto in questo programma che sfrutta pesantemente gli interrupt per la gestione della grafica. Un risultato apprezzabile può comunque essere ottenuto costringendo il 68000 a eseguire divisioni a vuoto: l'istruzione DIV è infatti la più lenta dell'intero set e non richiede accessi al bus. Durante l'esecuzione di questa istruzione il processore si mantiene lontano dal bus per circa 20 microsecondi e se la incorporiamo nel loop di attesa il Blitter ce ne sarà senz'altro riconoscente.

Luci e ombre

La direzione della fonte di luce è rappresentata in forma parametrica nelle variabili KX, KY e KZ; LK è invece il modulo del vettore risultante e viene calcolato da Length. Il calcolo dell'angolazione con cui la luce colpisce le varie facce viene svolto da WireShadow; se l'angolo è maggiore di novanta gradi allora la faccia non è illuminata, in caso contrario la sua luminosità deve essere proporzionale al coseno dell'angolo d'incidenza.

Il macrolinguaggio grafico

Non resta che giocherellare con il cubo e preparare un piccolo show. Se ogni volta che desideriamo far compiere al cubo un certo movimento fosse necessario richiamare l'apposita routine il programma ingrasserebbe a dismisura; in simili casi il metodo più compatto consiste nella creazione di uno pseudolinguaggio dotato di quei soli comandi che si prevede possano risultare utili nel corso dell'animazione. In questo caso i comandi, implementati tramite macro e interpretati dalla routine Mover, sono INIT, ZIP, REACH, REP, NEXT, PUSH, POP e QUIT.

INIT stabilisce la posizione iniziale del cubo, mentre ZIP è la direttiva più importante e deve essere seguita da otto parametri: un tempo, le tre componenti dell'accelerazione da usarsi nella traslazione, le corrispondenti tre componenti per la rotazione e un secondo tempo. Il primo specifica la durata, misurata in immagini, delle suddette accelerazioni mentre il secondo esprime il successivo intervallo di tempo durante il quale il cubo procederà per inerzia. REACH consente agli assi del cubo di assumere gradualmente e automaticamente una determinata inclinazione, mentre la coppia REP - NEXT (simile al FOR - NEXT del BASIC) consente un numero prefissato di iterazioni.

Il comando PUSH consente di congelare l'angolazione raggiunta dagli assi del nostro cubo e di far sì che ogni successiva rotazione venga considerata relativa alla posizione attuale; POP, com'è facilmente intuibile, neutralizza il PUSH più recente. QUIT infine provoca l'arresto del programma e il ritorno al CLI.

Tramite questi comandi è possibile descrivere degli spostamenti piuttosto complessi e la sequenza compresa nel listato, di nome Film, deve essere considerata a tal proposito un semplice esempio alterabile e arricchibile a volontà.

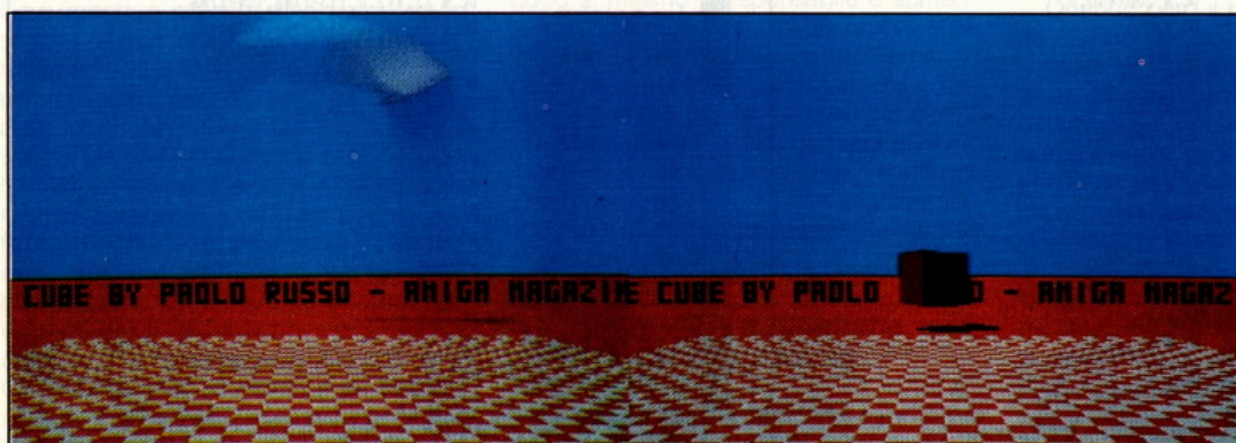
In conclusione

Il programma presentato, scritto con l'ausilio del macroassembler standard dell'Amiga prodotto dalla Metacomco, è in realtà molto complesso e nemmeno monopolizzando un intero numero della rivista sarebbe stato possibile illustrarne ogni singolo aspetto fin nei minimi dettagli. A ogni modo tutti i trucchi e le tecniche sfruttate in questo programma verranno esaurientemente descritte nel corso di Assembly, con particolare riferimento alle modalità di accesso al Blitter e al Copper; chi seguirà quel corso sarà poi in grado di scrivere autonomamente programmi di animazione paragonabili a questo se non migliori.

```
* The Cube by Paolo Russo - Amiga Magazine
            BRA MAIN
MIN         EQU 4
NT          EQU 40
LT          EQU 40
VX          DC.W 0
VY          DC.W 0
VZ          DC.W 0
VA          DC.W 0
VB          DC.W 0
VC          DC.W 0
XC          DC.W 0,0
YC          DC.W 100,0
ZC          DC.W 200+400,0
AA          DC.W 0
AB          DC.W 0
AC          DC.W 0
DIST        DC.W 250
YMAX        DC.W 200
KX          DC.W 50
KY          DC.W 100
KZ          DC.W 50
LK          DC.W 0
X1          DC.W 0
Y1          DC.W 0
X2          DC.W 0
Y2          DC.W 0
STACK       DC.L 0
PARAMS      DC.W 10238,0,16404
ALTERNATE   DC.W 10238,0,16404
```

```
NORMALS     DC.W 10238,0,16404
COLORS      DC.W 0,0,0
PLAN        DC.L 0,0,0,0,0,0,0
SIN         DC.L 0
PLANES      DC.L 0,0,0,0
PAGES       DC.L 0,0,0,0
PT          MACRO
            DC.W \1*100,\2*100,\3*100
            ENDM
LN          MACRO
            DC.W (\1-1)*4,(\2-1)*4;0
            ENDM
FC          MACRO
            DC.W \1,$\2,0,(\3-1)*6,(\4-1)*6
            DC.W (\5-1)*6,(\6-1)*6,(\7-1)*6,(\8-1)*6
            ENDM
P2D         DC.L 0,0,0,0,0,0,0,0
            DC.L 0,0,0,0,0,0,0,0
P3D         PT -1,-1,-1
            PT 1,-1,-1
            PT 1,1,-1
            PT -1,1,-1
            PT -1,-1,1
            PT 1,-1,1
            PT 1,1,1
            PT -1,1,1
LINES       LN 1,2
            LN 2,3
            LN 3,4
            LN 4,1
```

```
         LN 1,5
         LN 2,6
         LN 3,7
         LN 4,8
         LN 5,6
         LN 6,7
         LN 7,8
         LN 8,5
FACES    FC 1,F00,1,5,3,2,1,4
         FC 2,0F0,2,3,1,6,9,5
         FC 3,FFF,2,1,2,7,10,6
         FC 2,00F,3,2,3,8,11,7
         FC 3,FF0,1,2,4,5,12,8
         FC 1,F0F,5,1,9,10,11,12
TRIG     DC.W 0,402,804,1205,1606,2006,2404,2801
         DC.W 3196,3590,3981,4370,4756,5139,5519,5896
         DC.W 6270,6639,7005,7366,7723,8075,8423,8765
         DC.W 9102,9434,9760,10080,10394,10701,11003,11297
         DC.W 11585,11866,12139,12406,12665,12916,13159,13395
         DC.W 13623,13842,14053,14255,14449,14634,14811,14978
         DC.W 15137,15286,15426,15557,15678,15790,15893,15986
         DC.W 16069,16143,16207,16261,16305,16340,16364,16379,16384
COPLIST  DC.L $1002200
         DC.L $1020000
         DC.L $1040000
         DC.L $1080000
         DC.L $10A0000
         DC.L $0BE2C81
         DC.L $0902CC1
         DC.L $0920038
         DC.L $09400D0
         DC.L $0960020
         DC.L $180004C
CC1      DC.L $1820F00
CC2      DC.L $1840000
CC3      DC.L $1860FFF
         DC.L $1900A00
         DC.L $1920500
         DC.L $1940500
         DC.L $1960000
         DC.L $AC01FF00
         DC.L $1004600
         DC.L $1800A00
CCPLANE  DC.L $0E40000
```

```
         DC.L $0E60000
         DC.L $BC01FF00
         DC.L $1940AAA
         DC.L $1960555
         DC.L $FFFFFFFE
ZIP      MACRO
         DC.W \1
         DC.B \2,\3,\4,\5,\6,\7
         DC.W \8
         ENDM
REP      MACRO
         DC.W -1,\1
         ENDM
NEXT     MACRO
         DC.W -2
         ENDM
PUSH     MACRO
         DC.W -3
         ENDM
POP      MACRO
         DC.W -4
         ENDM
REACH    MACRO
         DC.W -5,\1,\2,\3
         ENDM
QUIT     MACRO
         DC.W -7
         ENDM
INIT     MACRO
         DC.W -6,\1,\2,\3,\4,\5,\6
         ENDM
FILM     REP -1
         INIT 0,100,600,0,0,0
         ZIP 100,0,0,0,0,0,0,0
         ZIP 40,0,-1,0,0,0,0,0
         ZIP 40,0,1,0,0,0,0,50
         ZIP 80,0,0,-1,0,0,0,0
         ZIP 80,0,0,1,0,0,0,25
         ZIP 160,0,0,1,0,3,0,0
         ZIP 160,0,0,-1,0,-3,0,0
         REACH 0,0,0
         ZIP 160,0,0,-1,3,0,0,0
         ZIP 160,0,0,1,0,0,0,0
         ZIP 80,0,0,0,-6,0,0,0
```

Evoluzioni in 3-D.

```
            REACH 8192,0,6420
            PUSH
            ZIP 256,0,0,0,0,3,0,0
            ZIP 64,-1,0,0,0,0,0,0
            ZIP 64,1,0,0,0,0,0,0
            REP 3
            ZIP 64,1,0,0,0,0,0,64
            ZIP 128,-1,0,0,0,0,0,64
            ZIP 64,1,0,0,0,0,0,0
            NEXT
            ZIP 32,4,0,0,0,0,0,0
            ZIP 32,-4,0,0,0,-24,0,0
            ZIP 256,0,0,0,0,16,0,100
            ZIP 256,0,0,0,0,-10,2,25
            ZIP 40,0,0,16,0,0,0,0
            ZIP 40,0,0,-16,0,0,0,0
            ZIP 16,-16,0,0,0,0,0,16
            ZIP 16,16,0,0,0,0,0,0
            ZIP 32,16,0,0,0,0,0,0
            REP 5
            ZIP 32,-16,-14,0,0,0,0,0
            ZIP 32,-16,14,0,0,0,0,0
            ZIP 32,16,14,0,0,0,0,0
            ZIP 32,16,-14,0,0,0,0,0
            NEXT
            ZIP 32,-16,0,0,0,0,0,0
            ZIP 40,0,0,-16,0,0,0,0
            ZIP 40,0,0,16,0,0,0,0
            ZIP 32,-16,0,0,0,0,0,0
            REP 5
            ZIP 32,16,-4,16,0,0,0,0
            ZIP 32,16,4,-16,0,0,0,0
            ZIP 32,-16,4,-16,0,0,0,0
            ZIP 32,-16,-4,16,0,0,0,0
            NEXT
            REP 4
            ZIP 32,16,-16,1,0,0,0,0
            ZIP 32,16,16,1,0,0,0,0
            ZIP 32,-16,16,1,0,0,0,0
            ZIP 32,-16,-16,1,0,0,0,0
            NEXT
            ZIP 48,16,0,0,0,0,0,16
            ZIP 16,-16,0,0,0,0,0,0
            ZIP 128,0,0,-8,0,0,0,84
            ZIP 512,0,0,1,0,0,0,100
            POP
            NEXT
            QUIT
C1          EQU CC1-COPLIST+652
C2          EQU CC2-COPLIST+652
C3          EQU CC3-COPLIST+652
CPLANE      EQU CCPLANE-COPLIST+652
GFXNAME     DC.B 'graphics.library',0
            CNOP 0,2
MAIN        MOVEM.L D0-D7/A0-A6,-(A7)
            MOVE.L 4,A6
            LEA PLAN(PC),A2
            MOVEQ #6,D2
1$          MOVE.L #10240,D0
            MOVE.L ##10003,D1
            JSR -198(A6) 'AllocMem
            MOVE.L D0,(A2)+
            DBEQ D2,1$
            BEQ.S 2$
            MOVE.L #3000,D0
            MOVEQ #3,D1
            JSR -198(A6) 'AllocMem
            MOVE.L D0,(A2)+
            BEQ.S 2$
            BSR.S CUBE
            MOVEQ #-1,D2
            LEA SIN(PC),A2
            MOVE.L (A2)+,A1
            MOVE.L #3000,D0
            JSR -210(A6) 'FreeMem
2$          NEG.W D2
            ADDQ.W #6,D2
            SUBQ.L #4,A2
            BRA.S 3$
4$          MOVE.L -(A2),A1
            MOVE.L #10240,D0
            JSR -210(A6) 'FreeMem
3$          DBRA D2,4$
            MOVEM.L (A7)+,D0-D7/A0-A6
            RTS
CUBE        LEA TRIG(PC),A0
            MOVE.L -(A2),A1
            LEA 650(A1),A3
            LEA 258(A1),A2
            MOVEQ #64,D0
1$          MOVE.W (A0)+,D1
            MOVE.W D1,(A1)+
            MOVE.W D1,510(A1)
            MOVE.W D1,-(A2)
            NEG.W D1
            MOVE.W D1,254(A1)
            MOVE.W D1,256(A2)
            DBRA D0,1$
            MOVE.L A3,A0
            LEA COPLIST(PC),A1
2$          MOVE.L (A1)+,D1
            MOVE.L D1,(A0)+
            ADDQ.L #2,D1
            BNE.S 2$
            MOVE.L A0,A5
            LEA P3D(PC),A1
            MOVEQ #11,D1
3$          MOVE.L (A1)+,(A0)+
            DBRA D1,3$
L1          LEA GFXNAME(PC),A1
            MOVEQ #0,D0
            JSR -552(A6) 'OpenLibrary
            MOVE.L D0,A6
            JSR -456(A6) 'OwnBlitter
            JSR -228(A6) 'WaitBlit
            BSR VECTOR
            MOVE.L (A0),4(A1)
            MOVE.L A1,(A0)
            CLR.B 3
            BSR LENGTH
            LEA PLANES+12(PC),A0
            MOVE.L PLAN+24(PC),(A0)
            BSR TILES
1$          BCLR #0,3
            BEQ.S 1$
            MOVE.L A3,$DFF080
            MOVE.W $DFF088,D0
            BSR MOVER
            MOVE.L 38(A6),$DFF080
            MOVE.W $DFF088,D0
            MOVE.W ##8020,$DFF096
            BSR.S VECTOR
            MOVE.L 4(A1),(A0)
            CLR.B 3
```

```
        JSR -462(A6) 'DisownBlitter
        MOVE.L A6,A1
        MOVE.L 4,A6
        JMP -414(A6) 'CloseLibrary
VECTOR  MOVEQ #0,D0
        MOVE.B -9,D0
        LSL.W #2,D0
        MOVE.L D0,A0
        LEA GEST(PC),A1
        RTS
PAGESWAP LEA PLANES(PC),A2
        LEA PLAN(PC),A1
        MOVEM.L 12(A1),D0-D2
        CMP.L (A2),D0
        BNE.S 1$
        MOVEM.L (A1),D0-D2
1$      EXG D1,D2
        MOVEM.L D0-D2,(A2)
        RTS
VIEW    MOVE.L PLANES+4(PC),A0
        LEA NORMALS(PC),A2
        BSR CLEAR
        BSR MAKESHADOW
        BSR PROJECT
        MOVE.L PLANES+8(PC),A0
        LEA PARAMS(PC),A2
        BSR CLEAR
        BSR SETPARAMS
        BSR WIRES
        BSR CALCPARAMS
        LEA PLANES(PC),A3
        LEA NORMALS(PC),A2
        MOVE.L (A3)+,A0
        BSR FILL
        LEA NORMALS(PC),A2
        MOVE.L (A3)+,A0
        BSR FILL
        LEA PARAMS(PC),A2
        MOVE.L (A3)+,A0
        BSR FILL
        BSR SWAPPARAMS
5$      BCLR #0,3
        BEQ.S 5$
        LEA PAGES(PC),A0
        MOVEM.L PLANES(PC),D0-D3
        MOVE.W #5120,A1
        ADD.L A1,D2
        MOVE.L SIN(PC),A2
        MOVE.W D2,CPLANE+4(A2)
        SWAP D2
        MOVE.W D2,CPLANE(A2)
        MOVE.L D1,D2
        ADD.L A1,D2
        ADD.L A1,D3
        MOVEM.L D0-D3,(A0)
        LEA COLORS(PC),A1
        MOVE.W (A1)+,C1(A2)
        MOVE.W (A1)+,C2(A2)
        MOVE.W (A1)+,C3(A2)
4$      BCLR #0,3
        BEQ.S 4$
        RTS
GEST    BRA.S 1$
2$      JMP 100000
1$      BTST #5,$DFF01F
        BEQ.S 2$
        ADDQ.B #1,3
        MOVEM.L D0-D3,-(A7)
        MOVEM.L PAGES(PC),D0-D3
        MOVEM.L D0-D3,$DFF0E0
        MOVEM.L (A7)+,D0-D3
        BRA.S 2$
CLEAR   MOVE.W #$0100,D0
        BRA.S BLITRECT
FILL    MOVE.W #$09F0,D0
BLITRECT LEA $DFF002,A1
        ADD.W (A2),A0
        MOVEQ #1,D1
        MOVEQ #6,D3
1$      DIVU D1,D1
        BTST D3,(A1)
        BNE.S 1$
        SUBQ.L #2,A1
        MOVEQ #-1,D1
        MOVE.W D0,64(A1)
        MOVE.W #$12,66(A1)
        MOVE.L D1,68(A1)
        MOVE.L A0,80(A1)
        MOVE.L A0,84(A1)
        MOVE.W 2(A2),100(A1)
        MOVE.W 2(A2),102(A1)
        MOVE.W 4(A2),88(A1)
        RTS
ROT     MOVE.W D1,D0
        MULS D4,D0
        MOVE.W D2,D6
        MULS D5,D6
        SUB.L D6,D0
        MULS D1,D5
        MULS D2,D4
        ADD.L D4,D5
        MOVE.L D0,D4
        ASL.L #2,D4
        SWAP D4
        ASL.L #2,D5
        SWAP D5
        RTS
ALPHA   MOVE.L SIN(PC),A0
        SWAP D0
        CLR.W D0
        ROL.L #8,D0
        ADD.W D0,D0
        ADD.W D0,A0
        SWAP D0
        LSR.W #1,D0
        MOVE.W (A0),D2
        MOVE.W 2(A0),D4
        MOVE.W 128(A0),D1
        MOVE.W 130(A0),D3
        SUB.W D2,D4
        ADD.W D4,D4
        MULS D0,D4
        SWAP D4
        ADD.W D4,D2
        SUB.W D1,D3
        ADD.W D3,D3
        MULS D0,D3
        SWAP D3
        ADD.W D3,D1
        LEA 48(A5),A0
        MOVEQ #7,D3
        RTS
ROTX    BSR.S ALPHA
1$      MOVE.W (A5)+,(A0)+
        MOVE.W (A5)+,D4
        MOVE.W (A5)+,D5
```

```
        BSR.S ROT
        MOVE.W D4,(A0)+
        MOVE.W D5,(A0)+
        DBRA D3,1$
        RTS
ROTY    BSR.S ALPHA
1$      MOVE.W (A5)+,D5
        MOVE.L (A5)+,D4
        BSR.S ROT
        MOVE.W D5,(A0)+
        MOVE.W -4(A5),(A0)+
        MOVE.W D4,(A0)+
        DBRA D3,1$
        RTS
ROTZ    BSR.S ALPHA
1$      MOVE.W (A5)+,D4
        MOVE.W (A5)+,D5
        BSR ROT
        MOVE.W D4,(A0)+
        MOVE.W D5,(A0)+
        MOVE.W (A5)+,(A0)+
        DBRA D3,1$
        RTS
PROJECT MOVE.L A5,A0
        LEA P2D(PC),A1
        MOVEQ #15,D3
FLAT    MOVEM.W (A0)+,D0-D2
        ADD.W ZC(PC),D2
        ADD.W DIST(PC),D2
        ADD.W XC(PC),D0
        MULS DIST(PC),D0
        DIVS D2,D0
        ADD.W #160,D0
        ADD.W YC(PC),D1
        MULS DIST(PC),D1
        DIVS D2,D1
        ADD.W #128,D1
        MOVE.W D0,(A1)+
        MOVE.W D1,(A1)+
        DBRA D3,FLAT
```

```
        RTS
WIRES   LEA FACES(PC),A1
        LEA LINES(PC),A2
        LEA P2D(PC),A3
        MOVEQ #11,D0
        LEA 4(A2),A0
1$      CLR.W (A0)+
        ADDQ.L #4,A0
        DBRA D0,1$
        MOVEQ #5,D0
2$      MOVE.W (A1)+,D7
        BSR WIRESHADOW
        MOVEM.W (A1),D1-D2
        MOVEM.W 0(A2,D1.W),D3-D4
        MOVEM.W 0(A2,D2.W),D5-D6
        CMP.W D4,D5
        BEQ.S 3$
        EXG D3,D4
        CMP.W D4,D5
        BEQ.S 3$
        EXG D5,D6
        CMP.W D4,D5
        BEQ.S 3$
        EXG D3,D4
3$      MOVEM.W 0(A3,D3.W),D1-D2
        MOVEM.W 0(A3,D4.W),D3-D4
        MOVEM.W 0(A3,D6.W),D5-D6
        SUB.W D3,D5
        SUB.W D1,D3
        SUB.W D4,D6
        SUB.W D2,D4
        MULS D6,D3
        MULS D5,D4
        MOVEQ #3,D1
        CMP.L D3,D4
        BLE.S 4$
        MOVEQ #0,D6
        MOVE.B D7,D6
        ADD.W D6,D6
        LEA COLORS(PC),A0
```

Ancora evoluzioni in 3-D.

```
                MOVE.W -6(A1),-2(A0,D6.W)
                BRA.S 5$
4$              CLR.B D7
5$              MOVE.W (A1)+,D2
                EOR.W D7,4(A2,D2.W)
                DBRA D1,5$
                DBRA D0,2$
                MOVE.L A2,A1
                MOVEQ #11,D6
6$              MOVEM.W (A1)+,D4-D5/D7
                MOVEM.W 0(A3,D4.W),D0-D1
                MOVEM.W 0(A3,D5.W),D2-D3
                LEA PLANES+8(PC),A2
                BSR.S DRAW
                MOVEM.W 32(A3,D4.W),D0-D1
                MOVEM.W 32(A3,D5.W),D2-D3
                LEA PLANES+16(PC),A2
                ROR.W #8,D7
                BSR.S DRAW
                DBRA D6,6$
RET             RTS
CUT             MACRO
                MOVEM.W \1-\2/D4-D5,-(A7)
                MOVE.W D4,\1
                MOVE.W D5,\2
                BSR.S OUTDRAW
                MOVEM.W (A7)+,D0-D3
                ENDM
DRAW            MOVEM.L D4-D6/A1/A3,-(A7)
                BSR.S DRAW2
                MOVEM.L (A7)+,D4-D6/A1/A3
                RTS
CUTF            CUT D2,D3
                BRA.S DRAW2
CUTL            CUT D0,D1
DRAW2           TST.B D7
                BEQ.S RET
                CMP.W D0,D2
                BNE.S 1$
                CMP.W D1,D3
                BEQ.S RET
1$              BSR.S FHI
SHUT            MACRO
                TST.W \3
                BMI.S OUTDRAW
                TST.W \2
                BPL.S SH1\@
                MOVEQ #0,\4
                BSR.S \1X
                BRA.S CUTF
SH1\@           MOVE.W #\5,\4
                CMP.W \4,\2
                BGT.S OUTDRAW
                CMP.W \4,\3
                BLE.S SH2\@
                BSR.S \1X
                BRA.S CUTL
SH2\@           CNOP 0,2
                ENDM
                SHUT HOR,D1,D3,D5,255
                BSR.S FLE
                SHUT VER,D0,D2,D4,319
                BRA.S INDRAW
CHECK           MACRO
                MOVE.W \1,D5
                BSR.S CHKLIM
                MOVE.W D5,\1
                ENDM
```

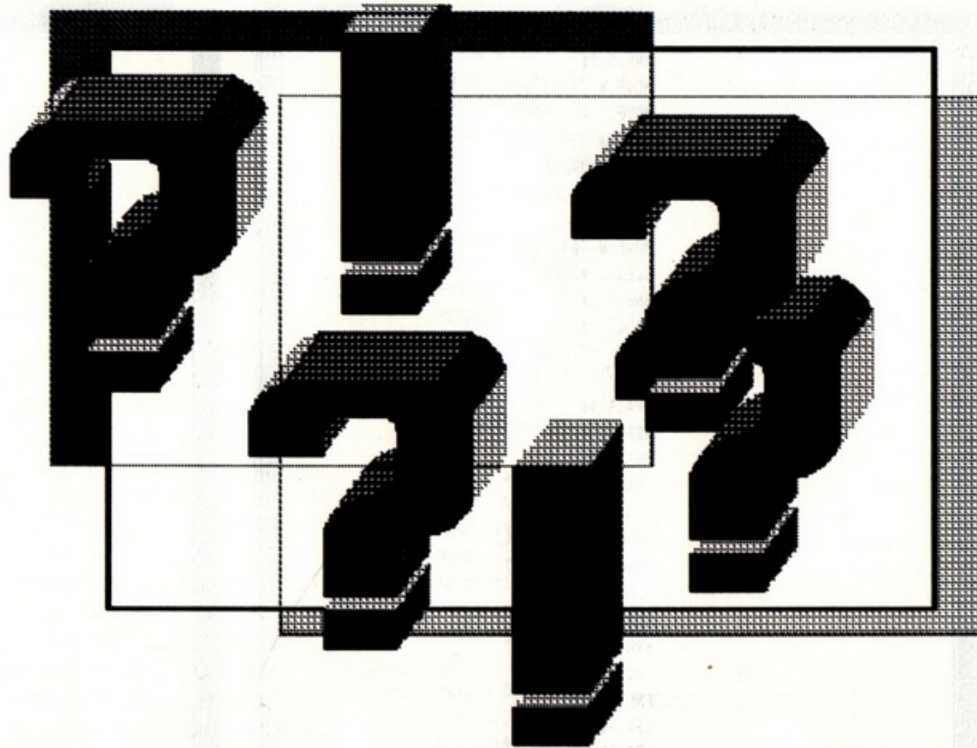
```
OUTDRAW         MOVE.L A2,-(A7)
                MOVE.W #255,D4
                CHECK D1
                CHECK D3
                MOVE.W #319,D4
                CHECK D0
                CHECK D2
                BSR.S INDRAW
                MOVE.L (A7)+,A2
                RTS
CHKLIM          TST.W D5
                BPL.S 1$
                MOVEQ #0,D5
1$              CMP.W D4,D5
                BLE.S 2$
                MOVE.W D4,D5
2$              RTS
FHI             CMP.W D1,D3
                BNE.S F1
FLE             CMP.W D0,D2
F1              BGT.S 1$
                EXG D0,D2
                EXG D1,D3
1$              RTS
XXX             MACRO
                MOVE.W \1,\5
                SUB.W \3,\5
                MOVE.W \6,D6
                SUB.W \2,D6
                MULS D6,\5
                MOVE.W \2,D6
                SUB.W \4,D6
                DIVS D6,\5
                ADD.W \1,\5
                RTS
                ENDM
HORX            XXX D0,D1,D2,D3,D4,D5
VERX            XXX D1,D0,D3,D2,D5,D4
INDRAW          BSR.S FHI
                BTST #7,D7
                BEQ.S L2
LOWER           MACRO
                CMP.W D4,\1
                BCC.S LOW\@
                MOVE.W \1,D4
LOW\@           CNOP 0,2
                ENDM
HIGHER          MACRO
                CMP.W \1,D4
                BCC.S HIG\@
                MOVE.W \1,D4
HIG\@           CNOP 0,2
                ENDM
                LEA X1(PC),A0
                MOVE.W (A0),D4
                LOWER D0
                LOWER D2
                MOVE.W D4,(A0)+
                MOVE.W (A0),D4
                LOWER D1
                LOWER D3
                MOVE.W D4,(A0)+
                MOVE.W (A0),D4
                HIGHER D0
                HIGHER D2
                MOVE.W D4,(A0)+
                MOVE.W (A0),D4
                HIGHER D1
```

```
             HIGHER D3
             MOVE.W D4,(A0)+
L2           SUB.W D0,D2
             SUB.W D1,D3
             MULU #40,D1
             MOVE.W D0,D4
             LSR.W #4,D4
             ADD.W D4,D4
             ADD.W D4,D1
             MOVEQ #15,D5
             AND.L D5,D0
             MOVE.W D0,D4
             ROR.L #4,D0
             NOT.W D4
             AND.W D5,D4
             MOVEQ #0,D5
             BSET D4,D5
             MOVE.W #4,D0
             TST.W D2
             BPL.S 1$
             ADDQ.W #1,D0
             NEG.W D2
1$           CMP.W D2,D3
             BLE.S 2$
             EXG D2,D3
             SUBQ.W #4,D0
             ADD.W D0,D0
2$           MOVE.W D3,D4
             SUB.W D2,D4
             LSL.W #2,D4
             ADD.W D3,D3
             MOVE.W D3,D6
             SUB.W D2,D6
             BPL.S 3$
             OR.W #16,D0
3$           ADD.W D3,D3
             LSL.W #2,D0
             ADDQ.W #1,D2
             LSL.W #6,D2
             ADDQ.W #2,D2
             SWAP D3
             MOVE.W D4,D3
             OR.L #$0B5A0003,D0
             LEA $DFF000,A0
             MOVE.W D6,A1
             MOVEQ #1,D4
5$           MOVE.L -(A2),A3
             BTST D4,D7
             BEQ.S 6$
             ADD.W D1,A3
             MOVEQ #-1,D6
4$           BTST #6,2(A0)
             BNE.S 4$
             EOR.W D5,(A3)
             MOVEM.L D0/D6/A3,64(A0)
             MOVEM.L A1/A3,80(A0)
             LEA 96(A0),A4
             MOVEQ #40,D6
             MOVE.W D6,(A4)+
             MOVE.L D3,(A4)+
             MOVE.W D6,(A4)+
             MOVE.L #$FFFF8000,114(A0)
             MOVE.W D2,88(A0)
6$           DBRA D4,5$
             RTS
SETPARAMS    LEA X1(PC),A0
             MOVE.L #$10001000,(A0)+
             CLR.L (A0)
```

```
             RTS
SWAPPARAMS   LEA PARAMS(PC),A0
             MOVEM.W (A0)+,D0-D2
             MOVEM.W (A0),D3-D5
             MOVEM.W D0-D2,(A0)
             MOVEM.W D3-D5,-(A0)
             RTS
CALCPARAMS   MOVEQ #40,D0
             MULU Y2(PC),D0
             MOVE.W X2(PC),D1
             LSR.W #4,D1
             MOVE.W X1(PC),D2
             LSR.W #4,D2
             NEG.W D2
             ADD.W D1,D2
             ADDQ.W #1,D2
             ADD.W D1,D1
             ADD.W D1,D0
             MOVEQ #-20,D1
             ADD.W D2,D1
             ADD.W D1,D1
             MOVE.W Y2(PC),D3
             SUB.W Y1(PC),D3
             ADDQ.W #1,D3
             LSL.W #6,D3
             OR.W D2,D3
             NEG.W D1
             LEA PARAMS(PC),A0
             MOVEM.W D0-D1/D3,(A0)
             RTS
MAKESHADOW   MOVE.L A5,A1
             LEA 48(A1),A0
             MOVEQ #7,D0
1$           MOVEM.W (A1)+,D1-D3
             ADD.W YC(PC),D2
             NEG.W D2
             ADD.W YMAX(PC),D2
             MOVEM.W KX(PC),D4-D6
             MULS D2,D4
             DIVS D5,D4
             ADD.W D4,D1
             MULS D2,D6
             DIVS D5,D6
             ADD.W D6,D3
             MOVE.W YMAX(PC),D2
             SUB.W YC(PC),D2
             MOVE.W D1,(A0)+
             MOVE.W D2,(A0)+
             MOVE.W D3,(A0)+
             DBRA D0,1$
             RTS
WIRESHADOW   MOVEM.W (A1)+,D1-D4
             MOVE.W #$8100,D6
             MOVE.L A1,-(A7)
             LEA KX(PC),A1
             LEA 0(A5,D3.W),A0
             LEA 0(A5,D4.W),A4
             MOVEQ #0,D3
             MOVEQ #2,D4
1$           MOVE.W (A4)+,D5
             SUB.W (A0)+,D5
             MULS (A1)+,D5
             ADD.L D5,D3
             DBRA D4,1$
             DIVS #200,D3
             EXT.L D3
             ASL.L #4,D3
```

```
        BPL.S 2$
        MOVEQ #0,D3
        MOVEQ #0,D6
2$      DIVU (A1)+,D3
        MOVE.L (A7)+,A1
        CMP.W #16,D3
        BLS.S 3$
        MOVEQ #16,D3
3$      CMP.W #MIN,D3
        BHI.S 5$
        MOVEQ #MIN,D3
5$      MOVEQ #0,D2
        ROR.W #8,D1
        MOVEQ #2,D4
4$      MOVE.B D1,D5
        EXT.W D5
        CLR.B D1
        ROL.W #4,D1
        MULU D3,D5
        MOVE.B D5,D2
        ROL.W #4,D2
        DBRA D4,4$
        ROL.W #8,D2
        AND.W #$FFF,D2
        MOVE.W D2,-6(A1)
        OR.W D6,D7
        RTS
LENGTH  LEA KX(PC),A0
        MOVEQ #0,D1
        MOVEQ #2,D0
1$      MOVE.W (A0),D2
        MULS (A0)+,D2
        ADD.L D2,D1
        DBRA D0,1$
        MOVEQ #15,D0
        MOVEQ #0,D2
2$      BSET D0,D2
        MOVE.W D2,D3
        MULU D3,D3
        CMP.L D1,D3
        BLS.S 3$
        BCLR D0,D2
3$      DBRA D0,2$
        MOVE.W D2,(A0)
        RTS
FLOOR   ADD.W DIST(PC),D1
        MOVE.W D1,D4
        MULS DIST(PC),D0
        DIVS D4,D0
        MOVE.W YMAX(PC),D1
        MULS DIST(PC),D1
        DIVS D4,D1
        ADD.W #160,D0
        ADD.W #128,D1
        RTS
DRAWFLOOR MOVEM.L D0-D7,-(A7)
        BSR.S FLOOR
        EXG D0,D2
        EXG D1,D3
        BSR.S FLOOR
        LEA PLANES+16(PC),A2
        MOVEQ #2,D7
        BSR DRAW
        MOVEM.L (A7)+,D0-D7
        RTS
TILES   MOVEQ #0,D0
        MOVEQ #0,D1
        MOVEQ #0,D2
        MOVEQ #0,D3
        BSR.S STRIPE
        MOVEQ #LT/2,D4
        MOVEQ #NT/2-1,D5
2$      SUB.W D4,D0
        SUB.W D4,D2
        MOVEQ #LT,D4
        BSR.S DRAWFLOOR
        DBRA D5,2$
        MOVEQ #0,D3
        BSR.S STRIPE
        MOVE.L PLANES+12(PC),A0
        MOVE.L A0,A4
        LEA NORMALS(PC),A2
        BSR FILL
        MOVE.W #255,D0
3$      LEA 40(A4),A1
        MOVEQ #9,D1
4$      MOVE.W (A4)+,D2
        MOVEQ #15,D3
5$      LSR.W #1,D2
        ADDX.W D4,D4
        DBRA D3,5$
        MOVE.W D4,-(A1)
        DBRA D1,4$
        LEA 20(A4),A4
        DBRA D0,3$
        MOVE.L PLANES+12(PC),A0
        LEA 5200(A0),A0
        LEA CPRT(PC),A1
6$      MOVE.W (A1)+,D4
        MOVEQ #15,D0
        MOVEQ #4,D1
7$      MOVEQ #$60,D2
8$      BTST D0,D4
        BEQ.S 9$
        OR.B D2,(A0)
        OR.B D2,40(A0)
9$      SUBQ.B #1,D0
        LSR.B #2,D2
        BCC.S 8$
        LEA 80(A0),A0
        DBRA D1,7$
        LEA -399(A0),A0
        BTST D0,D4
        BEQ.S 6$
        RTS
STRIPE  MOVEQ #NT-1,D4
1$      ADD.W #LT,D3
        BSR DRAWFLOOR
        DBRA D4,1$
        RTS
CPRT    DC.W $E924,$B7DA,$F3CE,0
        DC.W $F24E,$B6DE,$F70E,$F3CE,0
        DC.W $F7DE,$B524,0
        DC.W $F7C8,$F7DA,$F6DE,$924E,$F6DE,0
        DC.W $F7EA,$B6DE,$F39E,$F39E,$F6DE,0
        DC.W $0380,0
        DC.W $F7DA,$BEDA,$4924,$F2DE,$F7DA,0
        DC.W $BEDA,$F7DA,$F2DE,$F7DA,$E54E,$4924,$BFDA,$F3CF
MOVER   LEA STACK(PC),A0
        MOVE.L A7,(A0)
        LEA FILM(PC),A0
MOVLOOP MOVE.W (A0)+,D0
        BPL 1$
        ADDQ.W #1,D0
```

```
              BNE.S 2$
              MOVE.W (A0)+,D1
              MOVE.L A0,-(A7)
4$            MOVE.W D1,-(A7)
              BRA.S MOVLOOP
2$            ADDQ.W #1,D0
              BNE.S 8$
              MOVE.W (A7)+,D1
              SUBQ.W #1,D1
              BNE.S 3$
              ADDQ.L #4,A7
              BRA.S MOVLOOP
3$            MOVE.L (A7),A0
              BRA.S 4$
8$            ADDQ.W #1,D0
              BNE.S 9$
              LEA 144(A5),A5
              LEA AA(PC),A1
              MOVE.W 4(A1),-(A7)
              MOVE.L (A1),-(A7)
              CLR.L (A1)+
              CLR.W (A1)+
              BRA.S MOVLOOP
9$            ADDQ.W #1,D0
              BNE.S 10$
              LEA -144(A5),A5
              LEA AA(PC),A1
              MOVE.L (A7)+,(A1)+
              MOVE.W (A7)+,(A1)+
              BRA.S MOVLOOP
10$           ADDQ.W #1,D0
              BNE.S 15$
              BRA.S 14$
13$           MOVEQ #2,D3
              MOVE.L A0,A1
              LEA VA(PC),A2
              LEA AA(PC),A3
11$           MOVE.W (A1)+,D2
              MOVE.W (A2),D0
              MOVE.W (A3)+,D1
              BSR.S SEEK
              MOVE.W D0,(A2)+
              DBRA D3,11$
              BSR FRAME
14$           MOVEQ #2,D3
              MOVE.L A0,A1
              LEA AA(PC),A2
12$           CMPM.W (A1)+,(A2)+
              DBNE D3,12$
              BNE.S 13$
              ADDQ.L #6,A0
              BRA MOVLOOP
15$           ADDQ.W #1,D0
              BNE.S THEEND
              LEA VX(PC),A1
              CLR.L (A1)+
              CLR.L (A1)+
              CLR.L (A1)+
              MOVEQ #2,D0
16$           MOVE.W (A0)+,(A1)+
              CLR.W (A1)+
              DBRA D0,16$
              MOVE.W (A0)+,(A1)+
              MOVE.W (A0)+,(A1)+
              MOVE.W (A0)+,(A1)+
              BRA MOVLOOP
1$            MOVEQ #5,D2
              LEA VX+12(PC),A1
5$            MOVE.B 0(A0,D2.W),D1
              EXT.W D1
              ADD.W D1,-(A1)
              DBRA D2,5$
              BSR.S FRAME
              SUBQ.W #1,D0
              BNE.S 1$
              ADDQ.L #6,A0
              MOVE.W (A0)+,D0
              BRA.S 6$
7$            BSR.S FRAME
6$            DBRA D0,7$
              BRA MOVLOOP
THEEND        MOVE.L STACK(PC),A7
              RTS
SEEK          SUB.W D1,D2
              BPL.S 1$
              NEG.W D0
              NEG.W D2
              BSR.S 1$
              NEG.W D0
2$            RTS
1$            ADDQ.W #1,D0
              BMI.S 2$
              MOVE.W D0,D1
              MULU D1,D1
              EXT.L D0
              ADD.L D0,D1
              LSR.L #1,D1
              EXT.L D2
3$            CMP.L D1,D2
              BCC.S 2$
              SUB.L D0,D1
              SUBQ.L #1,D0
              BRA.S 3$
FRAME         MOVEM.L D0/A0,-(A7)
              BTST #0,$BFEC01
              BNE.S THEEND
              BSR PAGESWAP
              MOVE.L PLANES(PC),A0
              LEA NORMALS(PC),A2
              BSR CLEAR
              MOVEQ #2,D0
              LEA VX(PC),A0
              LEA XC(PC),A1
              LEA VA(PC),A2
              LEA AA(PC),A3
4$            MOVE.W (A0)+,D1
              SWAP D1
              CLR.W D1
              ASR.L #4,D1
              ADD.L D1,(A1)+
              MOVE.W (A2)+,D1
              ADD.W D1,(A3)+
              DBRA D0,4$
              MOVE.W AA(PC),D0
              BSR ROTX
              MOVE.W AB(PC),D0
              BSR ROTY
              MOVE.W AC(PC),D0
              BSR ROTZ
              BSR VIEW
              LEA -144(A5),A5
              MOVEM.L (A7)+,D0/A0
              RTS
              END
```

L'AmigaBasic consente al programmatore di crearsi delle routine di enorme utilità con il minimo sforzo. Una di queste è certamente quella che vi andremo a presentare in queste pagine.

UN UTILE REQUESTER

Arricchiamo l'AmigaBasic di un potente optional

di Alessandro Prandi

Un requester è un'area dello schermo, che presentandosi sotto forma di window, permette all'utente di rispondere ad una situazione verificatasi in un programma.

Le richieste che possono essere fatte vanno dalla semplice domanda di un file da caricare, all'informare l'utente di eventuali fatali errori verificatisi nell'esecuzione di un programma e comunque questa routine è utile in qualsiasi caso si presenti l'occasione di rispondere con un Sì o un No.

Il programma allegato a questo articolo vi permetterà sicuramente di aggiungere un tocco di classe ai vostri programmi.

I dilemmi... e i sottoprogrammi

Una finestra di requester viene generalmente usata per semplificare la risposta quando avviene una richiesta, essa comunque si presta in qualsiasi caso ci possano essere solo due risposte, positiva o negativa. Il programma stampato di seguito inizia dalla SUB REQUESTER STATIC e si conclude nella riga di END SUB. Tutte le righe di programma che ci sono prima fanno parte di un semplice demo per mostrare la semplicità d'uso di questa utility. Il listato come si presenta attualmente è un sottoprogramma, questo significa che sebbene sia contenuto nel lavoro di un programma più complesso quando sarà richiamato lavorerà in modo autonomo. Tutte le variabili contenute nel sottoprogramma sono riconosciute solo dallo stesso, e queste sono meglio conosciute come variabili 'locali'. Il fatto che più appare evidente da questa situazione è che questa utility può essere inserita in più programmi così come sta, senza per questo interferire con le variabili del programma principale nel caso ci siano delle omonimie.

Generalmente quando un sottoprogramma è stato scritto e corretto può essere salvato su disco e quindi aggiunto a qualsiasi altro programma che ne richieda l'uso. L'ideale quindi per un programma-

tore è incominciare a creare un'intera libreria di specifici sottoprogrammi e quindi semplicemente aggiungerli ai nuovi programmi che si stanno scrivendo. Questa tecnica di programmazione viene anche chiamata programmazione modulare. Il fatto di essere completamente indipendente certe volte causa degli inconvenienti, nel nostro caso c'è un canale di comunicazione delle informazioni tra sottoprogramma e programma tramite lo statement SHARED che si trova nella seconda riga. Ogni dato che passa al sottoprogramma o che da questo viene inviato al programma principale deve attraversare la lista di variabili contenuta nello statement SHARED, per cui ogni variabile così definita viene riconosciuta da entrambi. Nell'esempio, assieme al programma, tutte queste informazioni sono necessarie per rendere la funzione della richiesta a seconda della necessità.

I colori e Amiga

Ci sembra doveroso a questo punto spendere qualche parola sui numeri dei colori. I numeri che servono a selezionare i colori vanno dallo zero al tre e sono contenuti nell'esempio come standard usato dallo screen AmigaBasic. Se invece nelle vostre applicazioni usate uno standard diverso allora le scelte dei colori possono essere incrementate come si desidera. Il colore zero è il colore di default dello sfondo, (normalmente blu) e non può essere usato per scrivere i messaggi poiché avrebbero lo stesso colore dello sfondo.

Il sottoprogramma apre una finestra sullo schermo corrente e quindi vi direziona l'output, questo lavoro viene svolto dagli statement WINDOW e WINDOW OUTPUT.

Una volta direzionato l'output, tutte le istruzioni per la collocazione (LINE, LOCATE, PAINT) sono relative all'angolo alto sinistro della finestra e NON allo schermo sul quale la finestra è stata aperta. La collocazione della finestra all'interno dello schermo è specificata dalle variabili REQX1 e REQY. Dopo che il requester è stato disegnato la routine WAITER aspetta che il mouse sia posizionato su una delle due risposte, "SI" o "NO", e che venga premuto il tasto di selezione. Il sottoprogramma quindi riporta la risposta al demo tramite la variabile CHOICE$. Appena prima dell'uscita dalla routine il comando WINDOW CLOSE causa la chiusura della finestra cancellando tutti dati apparsi su di essa dallo schermo. L'aspetto più importante di questa tecnica della gestione delle finestre ora appare chiaro: le informazioni contenute nella schermata principale, le quali erano state coperte dal requester, vengono ridisegnate come prima, come se nulla fosse accaduto. Tutto questo viene svolto automaticamente dal sistema operativo.

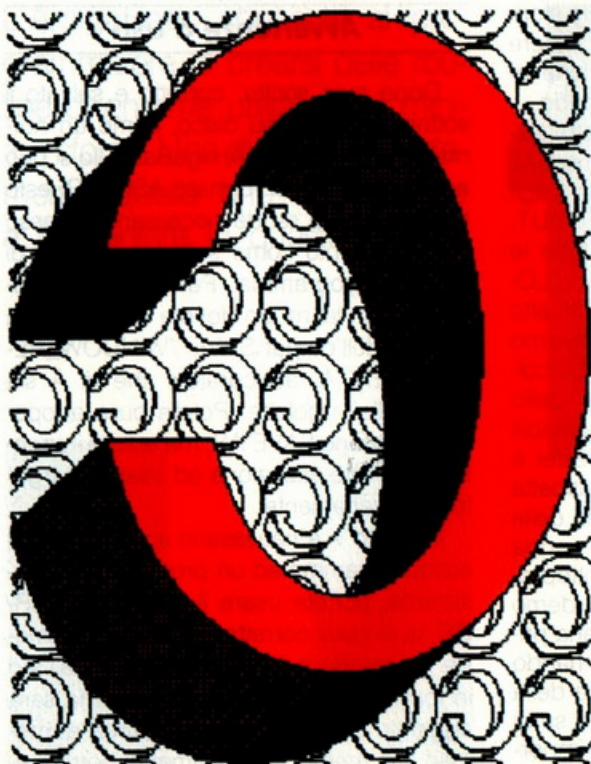
Avvertenze d'uso

Dopo aver scritto, corretto e salvato il sottoprogramma su disco, come un normale programma in AmigaBasic, lo si può anche salvare nel formato ASCII. Questo tipo di save si rende necessario se esso verrà aggiunto come sottoprogramma di un altro programma. Per salvare il programma in tal modo dovete scrivere nella finestra dell'output SAVE "WINDOW_REQUESTER",A, assicuratevi che la A sia fuori dalle virgolette. Potete pure adoperare l'opzione SAVE AS, ma anche in questo caso fate attenzione ad usare le virgolette correttamente.

Quando vi è necessario aggiungere un sottoprogramma ad un programma già esistente, potrete usare il comando MERGE, la sintassi corretta è: MERGE "nomefile", nomefile è il nome di un file salvato in formato ASCII. Il sottoprogramma sarà aggiunto alla fine del programma al momento in memoria. Ovviamente potrete rimuoverlo tramite le opzioni CUT e PASTE. Troppo facile, vero?

Speriamo proprio con questo piccolo listato di arricchire esteticamente e sopratutto in maneggevolezza e velocità i vostri programmi, e se così non fosse reclamate pure.

```
' Amiga Magazine
' Requester
  CLS
  \
INPUT "Indirizzo di partenza x"; reqx1
  INPUT "Indirizzo di partenza y"; reqy1
  INPUT "Colore dello sfondo (0-3)"; backcol
  INPUT "Colore del messaggio (0-3)"; msgcol
  INPUT "Colore delle linee esterne (0-3)"; outcol
  INPUT "Titolo"; title$
  INPUT "Messaggio stringa"; msg$
  CALL REQUESTER
  LOCATE 20,1: PRINT "Scelta utente...";CHOICE$
  STOP

SUB REQUESTER STATIC:
SHARED title$, msg$, reqx1, reqy1, backcol, msgcol,
                                   ?outcol, CHOICE$
  reqx2 = reqx1 + 206: reqy2 = reqy1 + 47
  yesx = 23: yesy = 26: nox = 134: noy = yesy
  WINDOW 2,title$,(reqx1,reqy1)-(reqx2,reqy2),0
  WINDOW OUTPUT 2: PAINT (100,20),backcol
```

```
  msgpad$ = " " + LEFT$ (msg$,22) + " "
  msglen = LEN (msgpad$)
  xloc = INT ((24-msglen)/2) + 1:xline = (xloc-1) * 8
  COLOR msgcol: LOCATE 2,xloc: PRINT msgpad$;
  LINE (xline,7)-(xline+8*msglen-1,7),0
  LINE (yesx,yesy)-(yesx+52,yesy+18),outcol,bf
  COLOR msgcol: LOCATE 5,5: PRINT " SI ";
  LINE (32,31)-(63,31),0
  LINE (nox,noy)-(nox+50,noy+18),outcol,bf
  LINE (144,31)-(175,31),0
  LOCATE 5,19: PRINT " NO ";

WAITER:
  CHOICE$ = "Nessuno"
  WHILE MOUSE (0) <>1
  WEND
  xpos = MOUSE (3): ypos = MOUSE (4)
  IF ypos < yesy OR ypos > yesy+18 THEN WAITER
  IF xpos >= yesx AND xpos <= yesx+54 THEN CHOICE$ = " SI "
  IF xpos >= nox AND xpos <= nox+48 THEN CHOICE$ = " NO "
  IF CHOICE$ = "Nessuno" THEN WAITER
  WINDOW CLOSE 2
  END SUB
```

APIRE E UTILIZZARE IL...

Ovvero come colmare una lacuna

di Mr. Lambda

La difficoltà maggiore che si incontra quando si desidera apprendere un linguaggio di una certa complessità e potenza sta nella carenza di documentazione e nella difficoltosa reperibilità di una versione aggiornata del linguaggio stesso. Noi intendiamo fare il possibile per mettervi a disposizione tutte quelle informazioni che vi permettano di introdurvi con facilità nell'ambiente e nel linguaggio C. Eccoci allora subito allo scopo di questo nostro primo appuntamento: muovere i primi passi nell'ambiente C, magari utilizzando il Lattice C 3.10.

Avvicinarsi al C, soprattutto per un possessore di Amiga, significa impadronirsi della possibilità di comprendere come utilizzare pienamente tutte le potenzialità offerte da questo potente computer e dal suo sistema operativo. È noto a tutti ormai che la maggior parte del sistema operativo e di tutto il software di utilità di Amiga è stato scritto in C. Le parti o sottosistemi di Amiga che non sono state scritte in C, sono state scritte in BCPL, un linguaggio che ha preceduto il C e di cui il C viene considerato un'evoluzione. Vi è comunque l'intenzione, per le prossime versioni del sistema operativo, di riscriverlo interamente in C.

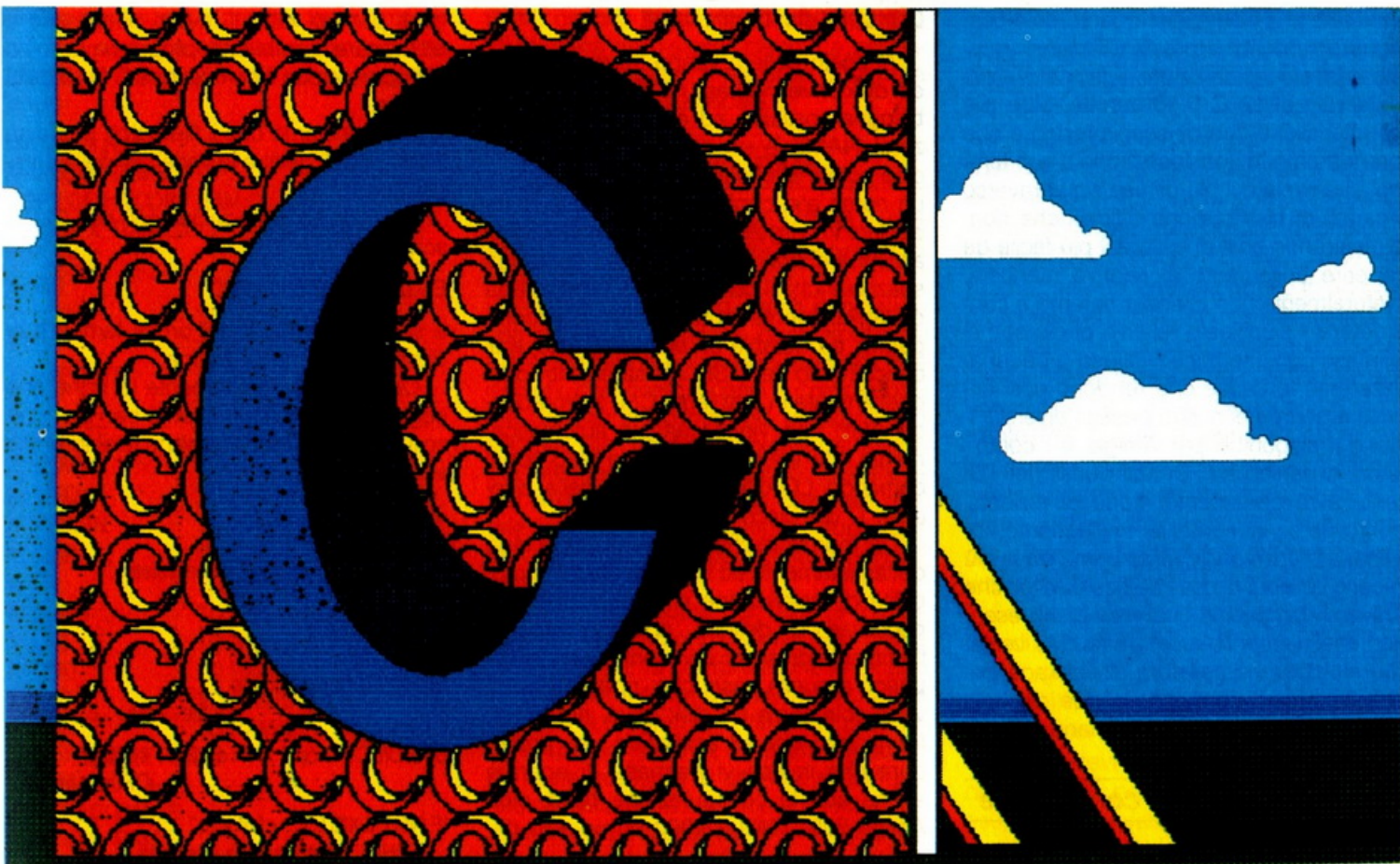
Ma capire il C, impadronirsene, è una fase indispensabile non solamente per utilizzare pienamente il sistema operativo di Amiga, ma anche, e soprattutto, per avvicinarsi a tutta quella letteratura di software che brulica e fermenta intorno a questa splendida macchina.

Prendiamo, come esempio, i Reference Manual: ad eccezione dello Hardware Reference Manual che tratta lo hardware in modo specifico e che utilizza l'assembler, i rimanenti manuali utilizzano estesamente ed efficacemente il linguaggio C. E già da questo fatto si può valutare l'importanza di essere in grado almeno di leggere e interpretare tale linguaggio. Ma proseguiamo. Guardiamo ora al Public Domain per Amiga; vera fucina di idee e di programmi. È un fatto che la stragrande maggioranza di esso è stato scritto e continua ad essere scritto in C. E ancora, con un' ottica più ampia che sa cogliere i mutamenti e gli sviluppi che avvengono nel mercato informatico, vogliamo un attimo ricordare l'ascesa irresistibile in atto di Unix?! Il suo nucleo centrale e la maggior parte del software in esso disponibile è scritto in C.

Non è certo questo il luogo per analizzare le ragioni storiche o formali dell'ascesa e dell'affermazione del linguaggio C, ma si può certamente indicare nella portabilità dei sorgenti, nella sinteticità delle sue espressioni, nella applicabilità generale e, non ultime, nell'efficienza e nella flessibilità che lo contraddistinguono, le cause e le caratteristiche che hanno portato al successo questo linguaggio. E vi possiamo assicurare che avrete modo di convincervene personalmente, se naturalmente ci seguirete in questa nostra serie di incontri.

Applicabilità

Si è parlato del C come di un linguaggio a medio livello, nel senso che possiede sia i potenti strumenti di programmazione strutturata propria anche di altri linguaggi definiti di alto livello, sia la possibilità di operare a basso livello, nel senso che può manipolare direttamente gli stessi oggetti trattati dal computer, cioè bit, numeri, caratteri e indirizzi. Ma anche dove, per flessibilità ed efficacia - di solito si tratta di soluzioni di problemi in cui il tempo è l'elemento critico - viene preferito l'assembly, il linguaggio C offre una capacità di interfacciamento organico e completo con esso. In alcuni compilatori, infatti, è presente la parola chiave o keyword asm che permette di introdurre direttamente nel codice sorgente istruzioni assembly, ottimi esempi dell'utilizzazione di questa modalità di scrittura di codice sono i compilatori Aztec, altri compilatori, come il Lattice C 3.10 che prenderemo in esame, effettuano il collegamento, con i codici oggetto generati dagli assembler, in fase di linking, cioè attraverso una particolare sintassi che viene interpretata durante l'esecuzione del pro-

gramma chiamato linker che effettua il collegamento dei moduli oggetto e la loro unificazione in un programma eseguibile.

Il Lattice 3.10 fornisce sia un comando LC, completo di tutte le opzioni per manipolare globalmente le operazioni di tutte le varie fasi del compilatore, e cioè LC1, LC2 e BLINK, sia il BLINK medesimo che è in grado individualmente di manipolare adeguatamente, ancora attraverso diverse opzioni, codici oggetto generati da assembler e dal compilatore per produrre programmi eseguibili.

A questo punto avrete senz'altro capito che il linguaggio C offre diversi livelli di approccio e di formalismo. E a tutti questi livelli corrispondono naturalmente gradi diversi di astrazione, isolamento, trasparenza che dir si voglia. Ma da che cosa, direte voi. Ma dallo hardware, Messieurs!

Alcune premesse indispensabili

Abbiamo scelto di dare a questi nostri articoli il taglio del work in progress, con riferimenti e interrelazioni costanti e graduali tra analisi di listati, grammatica del C e le diverse problematiche che possono sorgere in fase di progettazione, formalizzazione e/o debugging dei programmi. E proprio all'insegna del ' chi ben comincia e' a metà dell'opera ' che si e' preferito questo tipo di approccio e non la solita esposizione delle keyword, della sintassi e della programmazione strutturata.

Intendiamo muoverci, adeguatamente supportati dal materiale presente nel disco che accompagna la rivista, verso tre livelli di acquisizione: la capacità di lettura dei codici sorgenti, quindi la capacità di manipolarli, e infine lo sviluppo personale di programmi.

Abbiamo detto innanzitutto la capacità di lettura dei codici sorgenti, perché soprattutto coloro che programmano o preferiscono programmare in altri linguaggi come il BASIC, l'Assembly, il Modula-2, il Forth, il Lisp, siano in grado di trarre ispirazione e suggerimenti, per non dire soluzioni, dalla immensa letteratura disponibile in C. Quindi si è parlato di capacità di manipolazione dei codici sorgenti perché spesso la soluzione ai nostri problemi può essere trovata nei sorgenti già circolanti, magari di public domain, che però necessitano di essere manipolati adeguatamente per venire poi utilizzati nei nostri programmi. E proprio questa capacità di manipolazione vi sarà indispensabile se vorrete adeguare alle vostre necessità il software che volta per volta vi forniremo con il disco che accompagna la rivista.

E ogni nostro sforzo, infine, convergerà verso la programmazione vera e propria, quella cioè costituita da ideazione, algoritmo, formalizzazione e debugging, che dovrebbe mettervi nelle condizioni di realizzare ogni vostro progetto.

Questi diversi livelli interagiranno continuamente nell'articolazione del nostro discorso per produrre quello sviluppo progressivo delle conoscenze che vi porterà, speriamo, a quell'autonomia ed esperienza programmativa a cui tutti aspirano.

Una scelta preliminare

Oggi come oggi per Amiga sono disponibili sul mercato diverse versioni di due ottimi compilatori: Lattice e Aztec. Entrambi questi compilatori si presentano con potenti caratteristiche, migliorate e accresciute attraverso le diverse release rese disponibili di volta in volta. Ma noi abbiamo do-

vuto fare una scelta, per quanto dolorosa, riguardo al compilatore da utilizzare in queste nostre considerazioni. E la scelta è caduta sul Lattice C 3.10 perché, oltre alle caratteristiche che gli sono proprie e che prenderemo in considerazione a suo tempo, si è rivelato il più diffuso sia attraverso i canali di distribuzione ufficiali che non, dimostrando così di essere il più facile da reperire unitamente al relativo manuale. Naturalmente non ci soffermeremo a considerare il significato relativo di ciò che si è inteso con il termine ' diffuso ', ne' analizzeremo le cause che rendono così difficile reperire nel nostro paese i diversi linguaggi disponibili per Amiga, e i compilatori in particolare. Di tutti questi fatti l'utente avrà certamente modo di rendersi conto personalmente. Ci impegnamo, invece, a presentarvi quanto prima sia le più recenti versioni di casa Lattice e Aztec che i diversi compilatori che via via si presentino effettivamente sul mercato. Ma veniamo al nostro compilatore. Il package dovrebbe essere costituito da due dischetti da 3.5 pollici e dal relativo manuale ...

Come si utilizza il compilatore

Per ottenere gli aggiornamenti che riguardano la versione del compilatore in vostro possesso, potete selezionare l'icona ' Click For Info On Lattice C ', se naturalmente state utilizzando il Workbench, oppure potete editare il file READ.ME con TYPE se state utilizzando il CLI.

Il manuale della Lattice non dice nulla circa la possibilita' di utilizzare il compilatore con un solo disk drive, ma prevede solamente la possibilità di un suo utilizzo con configurazioni superiori, cioè con un secondo disk drive, o meglio, con un hard disk.

Per accontentare tutti coloro che posseggono un solo disk drive vi presenteremo un metodo di installazione che, senza rilevanti rinunce, vi metta in grado di utilizzare il compilatore.

In ogni caso la prima operazione da fare è la creazione di una copia di backup. Per effettuare questa operazione si possono utilizzare due metodi: il primo, sempre servendosi del Workbench, consiste nello spostare l'icona del disco sorgente sull'icona del disco destinazione, possibilmente vuoto.

Il secondo consiste nell'utilizzare il comando DISKCOPY attraverso l'item DUPLICATE del menu, oppure direttamente attraverso il CLI stesso:

```
DISKCOPY dfn: TO dfn: NAME
```

Al posto delle n vanno i numeri relativi dei disk drive, nel caso ne abbiate uno solo bisogna sostituire ad entrambe le n uno zero. NAME è un parametro opzionale, cioè potete anche trattenervi dall' attibuire un nome alla copia. Però ricordate di includere tra virgolette il nome che intendete assegnare alla copia nel caso contenga spazi bianchi. Il compilatore, come spiega dettagliatamente il manuale che lo accompagna, è bootable ed autosufficiente; e al caricamento del DISK # 1 sarete introdotti direttamente in ambiente CLI. Il Workbench infatti non viene caricato automaticamente, ma voi potete sempre farlo con il comando LOADWB presente nella directory C del disco stesso.

Dal momento che il manuale già spiega dettagliatamente ed in modo chiaro le modalità di installazione e gli assegnamenti necessari a definire appropriatamente l'ambiente di lavoro con configurazioni superiori a quella base, che, ripetiamo, prevede solamente un disk drive, noi ci soffermeremo ora ad analizzare una procedura che invece vi permetta di operare anche con la configurazione base, cioè Amiga 500, 1000 e 2000 ed un solo drive.

Senza trucco e senza inganno

Per poter predisporre un adeguato ambiente di lavoro, controllare efficientemente l'organizzazione dei file e gestire i vari dispositivi o device di AmigaDOS è indispensabile una buona conoscenza del CLI e dei comandi che esso mette a disposizione. Tale conoscenza è quindi presupposta se volete comprendere a fondo e personalizzare ciò che vi presentiamo, dal momento che ci serviremo proprio del CLI per predisporre il nostro ambiente di lavoro. (A coloro che avvertono la necessità di uno strumento che migliori il CLI in versatilità, potenza e comodità di utilizzo, consigliamo la Shell distribuita dalla Metacomco.) E ora veniamo ai file batch.

I file batch o file di comandi ci permettono di conservare dentro di essi una serie ordinata di comandi che quando inviamo al sistema operativo attraverso il comando EXECUTE, seguito dal nome del file batch e, opzionalmente da altri parametri, vengono eseguiti sequenzialmente dal computer.

Non vogliamo qui discutere dell'eleganza, praticità e potenza dei file batch, ma solamente utilizzarne qualcuno che svolgerà per noi tutto il lavoro necessario a ottimizzare l'ambiente programmativo in cui operare con il compilatore C 3.10 della Lattice.

Incominciamo con il file batch che avrà il compito di gestire le varie fasi della compilazione.

Innanzitutto introduciamoci nel CLI e scriviamo

```
ED DF0:S/MAKE
```

e saremo introdotti nell'ormai familiare editor di schermo di Amiga.

Ora copiate attentamente le righe che seguono e solo dopo averne ben compreso il significato modificatele in relazione alle vostre esigenze.

```
.KEY file,opt1,opt2,opt3
IF NOT EXISTS  <file> .c
 ECHO "Il file  <file> .c non esiste. "
   ECHO " "
   SKIP END
   ENDIF
ECHO "-- Compilazione
                 del file  <file> .c"

DF0:C/LC1  <opt1>  <opt2>  <opt3>
                       -iDF0:  <file>

IF NOT EXISTS " <file> .q"
  ECHO "Compilazione interrotta."
  QUIT 20
  ENDIF
ECHO " "
DF0:C/LC2  <file>

ASSIGN LIB: DF0:lib

ECHO " "
ECHO "-- Linking del file  <file> .o
         per creare il file  <file> "
ECHO "
" DF0/C/BLINK FROM LIB:c.o+  <file> .o
    TO  <file>  LIBRARY LIB:lc.lib +
                  LIB:amiga.lib
ECHO " "
ECHO "-- Linking del file ' <file> '
                  terminato. --"

LAB END
```

Memorizzate questo file nella directory S: semplicemente premendo prima il tasto ESC, che visualizzera' un'asterisco sulla linea dei comandi dell'editor, e poi il tasto della X seguito dal tasto del RETURN.

A questo punto ricominciamo le operazioni per scrivere un altro file batch. Questo avrà il compito di utilizzare oppurtunamente il device RAM: o disco RAM per predisporre il nostro ambiente di lavoro. Quindi digitiamo sempre nel CLI

```
ED DF0:S/CLI_TO_RAM
```

Ora continuate a copiare attentamente ciò che segue e altrettanto attentamente leggete le spiegazioni che vengono subito dopo:

```
MAKEDIR RAM:C
MAKEDIR RAM:S
COPY DF0:C/COPY RAM:C
ASSIGN X: RAM:C/COPY
CD SYS:C
X: DIR       TO RAM:C
X: LIST      TO RAM:C
X: CD        TO RAM:C
X: EXECUTE   TO RAM:C
X: ED        TO RAM:C
X: DELETE    TO RAM:C
X: STACK     TO RAM:C
X: INFO      TO RAM:C
X: ASSIGN    TO RAM:C
X: SKIP      TO RAM:C
X: IF        TO RAM:C
X: ECHO      TO RAM:C
X: LAB       TO RAM:C
X: QUIT      TO RAM:C
X: ENDIF     TO RAM:C
X: RUN       TO RAM:C
X: DF0:S/MAKE RAM:S
CD RAM:
ASSIGN C: RAM:C
ASSIGN S: RAM:S
```

Terminate anche questa sessione di editazione di file batch nello stesso modo che abbiamo visto sopra. Potete memorizzare questi file batch in tutti i dischi di cui intendete servirvi per memorizzare i programmi che svilupperete con il linguaggio C oppure in dischi che raccolgano i file di comandi o file batch che vi sono più utili. In ogni caso è vantaggioso che collochiate i file sempre nella directory S: dei dischi in cui intendete memorizzarli, perché, tra l'altro, il comando EXECUTE, quando si appresterà ad eseguirli, li cercherà di sicuro anche in quel luogo. Potreste pure, con il comando RENAME, rinominare il file CLI_TO_RAM chiamandolo STARTUP-SEQUENCE e lasciandolo sempre nella directory S: del disco. La conseguenza principale di questa vostra operazione sarà che al boot o avvio della macchina (COLDSTART), oppure al suo resettaggio (CTRL + tasto Amiga destro + tasto Amiga sinistro) verrà predisposto automaticamente l'ambiente di lavoro che vi permetterà di utilizzare il compilatore Lattice C 3.10 con la configurazione base, fornita cioè di un solo disk drive. Insomma dipende da come intendete utilizzare i vari dischi. Ricordate comunque che tutti i comandi, tutte le directory e tutti i file relativi ad esse che devono essere copiati nella RAM: devono essere NECESSARIAMENTE presenti nello stesso disco. La permalosità del computer al riguardo ci sempra sia pienamente giustificata.

Tutto dalla RAM

Una volta che il file CLI_TO_RAM è stato eseguito il comando delle operazioni si trasferisce alla RAM:.

A questo punto però, prima di poter incominciare a compilare, dobbiamo effettuare una fase di potatura. Sono infatti due i dischi che costituiscono il compilatore. Ma noi siamo costretti ad utilizzare uno soltanto. Ciò che vi proponiamo deve sempre intendersi come indicativo, le scelte finali si devono effettuare in base alle proprie esigenze, ma, comunque, dopo aver ben compreso il procedimento per giungervi.

Dalla nostra copia del DISK #2 cancelliamo la directory EXAMPLES, che è molto utile per la comprensione di alcune caratteristiche e per una esemplificazione di come si programma, ma poco per il lavoro, spesso ripetitivo, della compilazione. Se avete necessità di ulteriore spazio potete cancellare tutto ciò che vi sembra non strettamente indispensabile per la compilazione che vi apprestate a eseguire. Per esempio, soprattutto finché non avrete acquisito sufficiente dimestichezza con il linguaggio, potreste rinunciare anche alla directory SOURCE. In caso, quando vi sarà utile qualcosa, lo aggiungerete.

```
DELETE DF0:EXAMPLES
```

Ed ora aggiungiamo tutto ciò che ci può essere necessario e che è presente nel DISK #1. Siccome stiamo utilizzando la RAM: creiamo delle directory che ci semplifichino il lavoro. Inseriamo dunque il DISK #1 nel nostro drive e digitiamo:

```
MAKEDIR RAM:C
MAKEDIR RAM:L
CD DF0:C
COPY LC1 RAM:C
COPY LC2 RAM:C
COPY BLINK RAM:C
COPY ASM RAM:C
COPY OMD RAM:C
COPY OML RAM:C
CD DF0:L
COPY DISK-VALIDATOR RAM:L
CD RAM:
```

INSERITE IL DISK #2

```
MAKEDIR DF0:C
MAKEDIR DF0:L
COPY C DF0:C
COPY L DF0:L
```

Ricordate che almeno tre comandi devono essere copiati; e cioè LC1, LC2 e BLINK. Infatti questi comandi, oltre a comparire nel file MAKE che effettua le varie fasi della compilazione automaticamente, costituiscono le diverse fasi di una compilazione completa: espansione di macro, analisi sintattica, generazione di codice - LC1 e LC2 -, e linking - BLINK -, cioè trasformazione del codice oggetto in un programma eseguibile. Anche LC effettua le varie fasi della compilazione automaticamente, ma necessita di una particolare sintassi e non sempre la memoria vi è sufficiente. Per tutti questi comandi, come per le opzioni che sono disponibili è indispensabile comunque consultare il manuale. Prima di effettuare una scelta dovete sempre essere consapevoli di ciò che questa vostra scelta comporti.

La dimensione dei file dei diversi comandi vi costringerà a dover togliere il disco sorgente, cioè la copia del DISK #1, e sostituirlo con il disco destinazione, cioè la copia del DISK #2, più volte. Magari accontentatevi di memorizzare nella RAM: due o tre file per volta. State sicuri che il risultato vale la fatica. Una volta terminate tutte le operazioni necessarie, siete in possesso di un compilatore che risiede su di un unico disco.

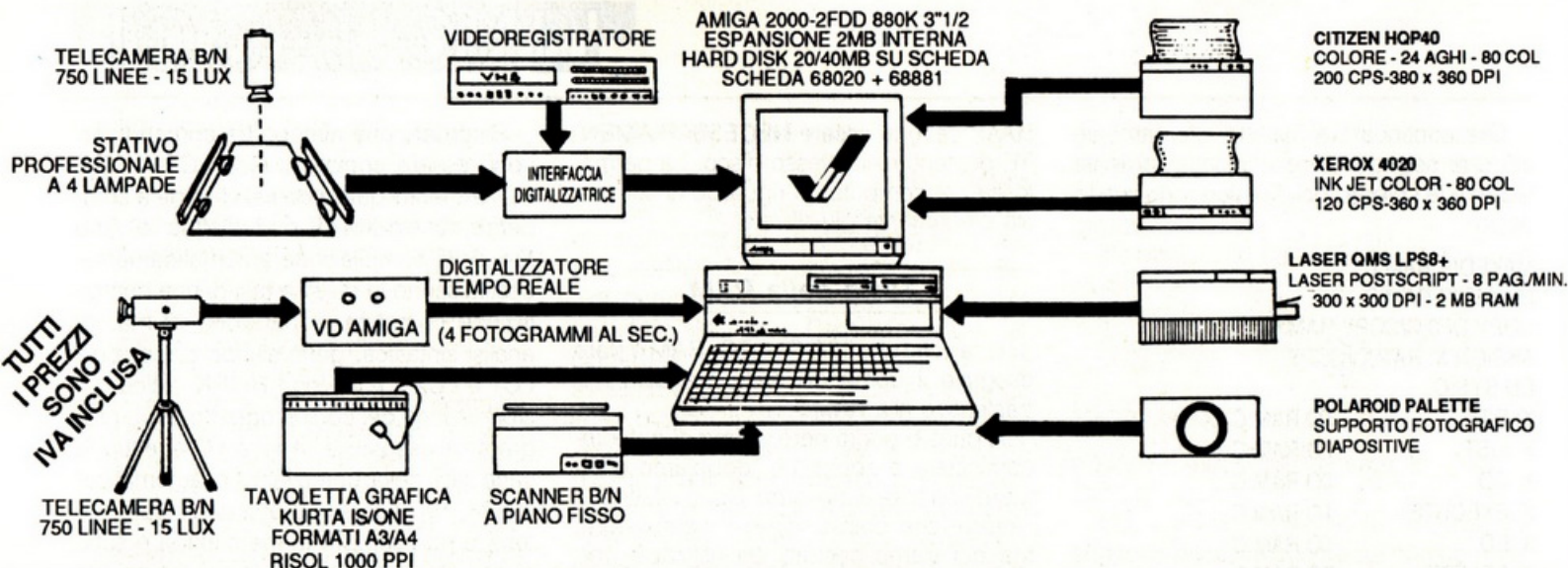
Da questo momento con ED potete scrivere i vostri file sorgenti in C memorizzandoli nella RAM:; oppure con COPY potete copiare, sempre nella RAM:, i file sorgenti che volete compilare. La cosa fondamentale, come avrete capito, è che i sorgenti risiedano nella RAM:. A quel punto basta digitare

```
EXECUTE MAKE nome_del_file_sorgente
```

seguito eventualmente dalle opzioni che vi interessano. Si ricordi che il nome del file sorgente, quando viene memorizzato nella RAM:, DEVE terminare con l'estensione .c - per esempio game.c - mentre, quando viene utilizzato nella linea di comandi per la compilazione vista sopra, NON deve terminare con l'estensione .c

```
EXECUTE MAKE game
```

A prima vista tutto questo sempra un po' laborioso, poi invece è laborioso veramente; ma, come abbiamo detto, i risultati vi compenseranno della fatica. È certo che quando terminerete il vostro lavoro saprete molto di più sul CLI e sul vostro compilatore Lattice C 3.10. Buon lavoro.

AMIGA WORKSTATION GRAFICHE

HARDWARE

AMIGA 500 .... 930.000
AMIGA 500 + Monitor 1084 .... 1.550.000
AMIGA 2000 senza monitor 1.950.000
AMIGA 2000 2 drive 3"1/2 .... 2.190.000
ESPANSIONE 512K interna A500 .... Telef.
ESPANSIONE 1MB esterna A1000 .... Telef.
ESPANSIONE 2MB esterna A500/A1000 .... Telef.
ESPANSIONE 2MB interna A2000 .... Telef.
DISK DRIVE 3"1/2 esterno A500/A1000 .... 290.000
DISK DRIVE 3"1/2 interno A2000 .... 250.000
HARD DISK 20MB EST. A500/A1000 .... 1.250.000
HARD CARD 20MB SCSI A2000 .... 1.250.000
HARD CARD 20MB 20MB SCSI A2000 .... 750.000
HARD CARD 40MB MS-DOS A2000 .... 950.000
Sistema a Cartridge da 12MB removibili della Kodak + 5 Cartridge (60 MB) 2.950.000
SCHEDA JANUS XT A2000 .... 850.000
SCHEDA JANUS AT A2000 .... 1.550.000
KIT SOSTITUZIONE MOTOROLA 68010 .... 99.000
SCHEDA 68020 + 68881 16MHZ 1.850.000
AMIGA-EYE A500/A1000/A2000 .... 130.000
VD AMIGA FRAMEGRABBER 750.000
VD 2000 DIGITALIZZATORE COLORE IN CVBS A500/A1000/A2000 .... 1.150.000
TELECAMERA SECURIT T-979

STATIVO PROFESSIONALE 4 LAMPADE 350.000
AMIGA SOUND A500/A1000/A2000 .... 150.000
INTERFACCIA MIDI A500/A1000/A2000 .... 99.000
GENLOCK PROFESSIONALE 850.000

TAVOLETTE GRAFICHE KURTA:
PENMOUSE (6" x 3" 200 PPI) 250.000
SERIE IS 8,5" x 11" 1000 PPI 790.000
SERIE IS 12" x 12" 1000 PPI 990.000
SERIE IS 12" x 17" 1000 PPI 1.690.000
PENNA A DUE BOTTONI 290.000
CURSORE A 4 BOTTONI 290.000
CAVO E SOFTWARE PER AMIGA 110.000

STAMPANTI:
PANASONIC KX-P1081 80 COL 120 CPS 550.000
NEC P2200 80 COL 216 CPS 24 AGHI 950.000
NEC P6 80 COL 216 CPS 24 AGHI .... Telef.
NEC P6 KIT COLORE .... Telef.
NEC P7 136 COL 216 CPS 24 AGHI 1.650.000
NEC P7 136 COL 216 CPS 24 AGHI 1.790.000
CITIZEN HQP40-24 AGHI 1.350.000
CITIZEN HQP40-KIT COLORE 1.550.000
XEROX 4020 INK JET COLORE 3.450.000
OKI LASER LL6 PPM 3.850.000
LASER QMS LPS8+POSTSCRIPT .... Telef.
HARD COPIER SHINKO .... Telef.
POLAROID PALETTE PER AMIGA 3.450.000

SOFTWARE ORIGINALE:

INFINITY SOFTWARE:
SHAKESPEARE 289.000
GALILEO 2.0 .... 89.000
ISM INC:
THE SURGEON 65.000
MICROSEARCH:
CITY DESK 189.000
MICROPROSE:
SILENT SERVICE 55.000
MOEBIUS 49.000
ULTIMA III 49.000
MICROMAGIC:
FORMS IN FLIGHT 110.000
MICROILLUSIONS:
FIRE POWER 35.000
DYNAMIC CAD 690.000
PHOTO PAINT 380.000
35.000
MINDSCAPE:
DEFENDER OF THE CROWN 59.000
HALLEY PROJECT 69.000
DEJA VU 69.000
UNINVITED 69.000
NEWTEK:
DIGI-PAINT 79.000
OXXY INC:
MAXIPLAN 500 .... 190.000
MAXIPLAN PLUS. 250.000
PSYGNOSIS:
BARBARIAN 55.000
OBLITERATOR 55.000
SUBLOGIC:
FLIGHT SIMULATOR 75.000
JET 75.000
SCENERY DISK 7 .... 39.000
ZUMA:
TV SHOW 129.000
GOLD DISK:
PROFESSIONAL PAGE 445.000
PAGESETTER ITAL. 210.000
ACTIVISION:
HACKER II 29.500
THE ART OF CHESS 29.500
SHANGHAI 29.500
BORROWAD TIME 65.000
LITTLE COMPUTER PEOPLE 35.000
MINDSHDOW 35.000
TASS TIMES ....
PORTAL 55.000
GEE BEE AIR RALLY 55.000
AEGIS:
ANIMATOR 175.000
ARAZOK'S TOMB 49.000
AUDIOMASTER 75.000
DIGA 99.000
DRAW PLUS 320.000
IMPACT 110.000
SONIX 99.000
VIDEOTITLER 125.000
PORT OF CALL 129.000
VIDEOSCAPE 3D 299.000
BYTE BY BYTE:
SCULPT 3D 129.000
ANIMATE 3D 199.000

COMMODORE:
MIND WALKER 69.000
TEXTCRAFT PLUS 145.000
SUPERBASE PERSONAL 190.000
LOGISTIX 120.000
DISCOVERY:
ARKANOID 75.000
EPYX:
DESTROYER 29.000
WINTER GAMES 29.000
WORLD GAMES 29.000
NEW HORIZONS:
PROWRITE 175.000
NORTHEASTERN SOFT:
PUBLISHER PLUS 129.000
RIGHT ANSWER GROUP:
THE DIRECTOR 89.000
METACOMCO:
MCC PASCAL 139.000
ASSEMBLER LANGUAGE 139.000
EAGLE SOFTWARE:
BUTCHER 2.0 .... 49.000
ELECTRONICS ARTS:
ADVENTURE C. SET 38.000
ARTIC FOX 29.500
BARD'S TALE I 29.500
CHESSMASTER 2000 .... 29.500
INSTANT MUSIC 33.000
MARBLE MADNESS 29.500
SKYFOX 29.500
TEST DRIVE 33.000
DE LUXE MUSIC C.S. 94.000
DE LUXE PAINT II 99.000
DE LUX PRINT 90.000
DE LUXE VIDEO 1.2. .... 109.000
FERRARI FORMULA 1 .... 38.000
RETURN TO ATLANTIS 38.000
PROGRESSIVE P. &S:
PIXMATE 94.000
MASTERTRONIC:
BLASTABALL 19.900
FEUD 19.900
KIKSTART II 19.900
NINJA MISSION 19.900
SPACE RANGER 19.900
FIREBIRD:
BUBBLE BOBBLE 29.000
MIRRORSOFT:
DARK CASTLE 49.000
KING OF CHICAGO 59.000
TETRIS 39.000
ANCO:
DEMOLITION 19.900
FLIGHT PATH 737 .... 19.900
GRID START 19.900
JUMP JET 19.900
KARTING GRAND PRIX 19.900
LAS VEGAS 19.900
PHALANX 19.900
SKY FIGHTER 29.000
STRIP POKER 19.900
THAI BOXING 19.900
XR 35 .... 19.900
RAINBIRD:
DRUM STUDIO 79.000
GOLDEN PATH 79.000
JINXTER 49.000
CDS:
FOOTBALL FORTUNE 49.000
MELBOURNE HOUSE:
ROADWARS 39.000
XENOX 39.000

PERSONAL COMPUTER

LINEA HITECH PERSONAL COMPUTER

LINEA XT 4.7/10 MHZ
XT-HT 256K 1FDD 360K TAST. AVANZ. 850.000
XT-HT 256K 2FDD 360K TAST. AVANZ. 1.050.000
XT-HT 256K 1FDD 360K HD 20MB TAST. AVANZ. 1.550.000

LINEA AT 10MHZ 0 WAIT STATE
AT-HT 512K 1FDD 1.2MB TAST. AVANZ. 1.950.000
AT-HT 512K 1FDD 1.2MB 1 HD 20MB TAST. AVANZ. 2.550.000
AT-HT 512K 1FDD 1.2MB 1 HD 85MB TAST. AVANZ. 3.150.000
AT-HT 512K 1FDD 1.2MB 1 HD 140MB TAST. AVANZ. 4.750.000

LINEA 386 16-20 MHZ
TOWER 2MB 1FDD 1.2MB 1 HD 40MB TAST. AVANZ. 6.380.000
TOWER 2MB 1FDD 1.2MB 1 HD 85MB TAST. AVANZ. 7.750.000
TOWER 2MB 1FDD 1.2MB 1 HD 140MB TAST.AVANZ. 9.850.000

SCHEDE PC
SCHEDA SERIALE 58.000
SCHEDA PARALLELA CENTRONICS 36.000
SCHEDA EGA AUTOSWITCH 490.000
SCHEDA FAX 1.450.000
SCHEDA COPY CARD II 160.000

HARD DISK
HARD DISK 20MB + CONTROLLER 590.000
HARD DISK 40MB + CONTROLLER 950.000
HARD CARD 20MB 690.000
HARD CARD 40MB 1.050.000

COPROCESSORI MATEMATICI
INTEL 8087 6MHZ 250.000
INTEL 8087 8MHZ 380.000
INTEL 80287 6MHZ 390.000
INTEL 80287 8MHZ 580.000
INTEL 80287 10MHZ 690.000
INTEL 80387 16MHZ 1.250.000

MONITOR
PHILIPS 7502/7513 MONOCROMATICO 12". 180.000
PHILIPS 9073 EGA COLORE 14" 850.000
PHILIPS 8833 COLORE 14" 550.000
MULTISYNC MONOCROMATICO 550.000
MULTISYNC COLORE 1.250.000

MODEM
ESSEGI 1200M 300/1200 BAUD V21/V22 FULL DUPLEX 360.000
ESSEGI 1203M 300/1200/75 V21/V23 VIDEOTEL 420.000
ESSEGI 2400M 1200/2400 BAUD V22/V22 BIS 750.000
ESSEGI 1200C CARD 360.000

TELEFAX
TELEFAX BACON-TELEFONO G2/G3 FORMATO A4 .... 2.250.000

PIX COMPUTER S.R.L.
VIA F. D'OVIDIO, 6C
TEL. 06/8293507-825731
00137 ROMA
COMPUTER & Co.
P. IVA 08309630583

DISPONIBILE LATTICE C COMPILER VERS. 40 LIT. 450.000

VENDITA PER CONTRASSEGNO SU TUTTO IL TERRITORIO NAZIONALE. OFFERTE E PREVENTIVI SU WORKSTATIONS GRAFICHE COMPLETE. SETTORI CAD 2D/CAD 3D/ANIMAZIONI 3D/DIGITALIZZIONI/VIDEO BROADCAST/DESKTOP PUBLISHING.
SI INVIANO A RICHIESTA SCHEDE TECNICHE PRODOTTI.
SCONTI PER RIVENDITORI QUALIFICATI.

— **Dedicato a Disk Magazine**

— **WITZ**

Tfish
Blobs

— **GIOCHI**

Reversi
FastLife

— **MAGAZINE**

— **GRAFICA**

ColorArt
Shuttle

— **STRUMENTI**

M.B.
Driver-stampanti

— **MUSICA**

Digital Filter

Dedicato a Disco Magazine

Avvertenza:
Il Disco-Magazine-team si riserva di apportare tutte le modifiche al testo che ritiene irrinunciabili e costituzionali, e che abbiano lo scopo di migliorare e/o aggiornare il prodotto o renderlo competitivo, senza avviso alcuno, anche durante l'acquisto della rivista e/o durante la sua lettura.

Mi trovavo in una Tipografia dell'Inferno, e vidi il metodo con cui il sapere è trasmesso da generazione in generazione.
Nella prima stanza c'era un Drago-Uomo che sgomberava dalla bocca della caverna i detriti; dentro numerosi Draghi ne proseguivano lo scavo.
Nella seconda stanza c'era una Vipera attorcigliata alla roccia e alla caverna, mentre altre erano intente ad adornarla con oro, argento e gemme.
Nella terza stanza c'era un'Aquila le cui ali e le penne erano d'aria; e per questa causa l'interno della caverna era infinito: intorno, numerose Aquile simili a uomini erigevano palazzi sulle rupi immense.
Nella quarta stanza c'erano Leoni di fuoco fiammeggiante, che aggirandosi rabbiosi fondevano i metalli in fluidi viventi.
Nella quinta stanza c'erano forme Innominate, che spargevano i metalli fusi nello spazio.
Dove, raccolti da Uomini che occupavano la sesta stanza, prendevano forma di libri, che poi venivano ordinati in biblioteche.

MEMORABILE APPARIZIONE,
William Blake

di Mr. Lambda

È qui tra noi.

E questo è il tempo del suo pieno dispiegarsi.

Come per ogni cosa, anche per questo il tempo è venuto.

Lasciatevi condurre nell'Evento Amiga affinché la sua magia penetri in voi e accompagni la vostra mente.

" Non abbiamo dato retta a storie ingegnosamente inventate... ne siamo stati testimoni oculari. " Pietro 1,16

Siamo qui per incuriosirvi, per divertirvi, per sbalordirvi. In verità una passione per questa macchina ci brucia. Saranno i suoi modi. Quella sua particolare maniera di assecondarci e farci sognare. Quindi non esitiamo nel dirvi subito che il suo non è stato un arrivo, un presentarsi, ma un incedere.

Nel tempo lontano del suo avvento, Amiga sollevò molte perplessità per quanto riguardava la disponibilità del software che mettesse in piena evidenza le straordinarie doti della sua architettura hardware e software. Tutti questi dubbi sono svaniti. Oggi il software disponibile è così numeroso e vario che è difficile perfino effettuarne un controllo sistematico.

Abbiamo l'onore di introdurvi nell'immenso universo di idee che circonda l'Amiga. E'speriamo che voi vi troviate sollecitazioni, stimoli e divertimento. Certi che è nel potere di questa macchina sorprendervi e ammaliarvi, ci presentiamo come i custodi di tutti quei fermenti che fanno di questa macchina un potente strumento di creatività e lavoro. La maggior parte del materiale che vi presenteremo proviene naturalmente dal Public Domain, ma il disco conterrà anche i programmi presenti nella rivista e tutti quei programmi che riterremo validi per la divulgazione. Il nostro fine è la diffusione di tutto ciò che possa aumentare la conoscenza di Amiga e ne possa mettere in luce le straordinarie possibilità.

Il disco che vi presentiamo in questa occasione si articola in cinque directory:

Witz
Giochi
Musica
Grafica
Strumenti
Magazine

Questa organizzazione del materiale in sei directory così chiamate è dovuta solamente ad una provvidenziale intuizione, dopo lunghi periodi di veglia notturna. Ancor oggi, infatti, il loro senso ci si impone così ricco di latenti e raffinati significati da sembrare quasi autoesplicativo. Insomma, i sortilegi dovrebbero pur riuscire pienamente incomprensibili.

WITZ è la directory delle curiosità e delle sorprese, delle spiritosaggini e delle trovate. Per farvi assaporare a fondo il piacere che accompagnerà la visione di questi programmi, vi comunichiamo fin d'ora che i relativi file sorgente tengono una compagnia discreta ma rassicurante ai programmi eseguibili. Per accedervi basta che chiamiate il CLI.

Non poteva mancare in tanta spettacolarità l'aspetto ludico. Ed eccovi subito serviti! Vi presentiamo la directory GIOCHI che non mancherà di stuzzicare la vostra curiosità e il vostro spirito di competizione. Sappiate fin d'ora che per quanti sforzi siano già stati fatti non si è ancora riusciti a programmare la magnanimità. Questo è il motivo per cui vi consigliamo di spremere da voi stessi tutta l'intraprendenza, l'astuzia e l'intuito di cui siete capaci, se volete cimentarvi con il nostro computer, oppure di cambiare directory.

La directory MUSICA contiene una sorpresa che interesserà tutti coloro che vogliono approfondire la conoscenza del mondo sonoro. E non fateci dire altro.

E veniamo alla directory GRAFICA. Certamente si poteva fare di più, e di più verrà certamente fatto prossimamente, ma si è voluto incominciare con questi due interessanti programmi in AmigaBASIC per offrire la possibilità di manipolare facilmente la stupenda e flessible grafica di Amiga.

Molti utenti avranno avuto qualche difficoltà ad utilizzare le loro stampanti con Amiga, probabilmente per la mancanza del driver adeguato. Ecco allora che la directory STRUMENTI potrebbe contenere qualcosa per loro. Ma non basta, sempre in questa directory potrete trovare un pratico indicatore di memoria.

E ora veniamo a coloro che, per pigrizia, non avranno voglia di digitare i programmi pubblicati sulla nostra rivista, ma saranno ben lieti di utilizzarli; oppure a coloro che vogliono accertarsi della correttezza dei listati pubblicati. Per tutti questi lettori e anche per tutti gli altri, la directory MAGAZINE contiene i listati dei programmi che compaiono nella rivista.

Ed ora soffermiamoci su alcune importanti convenzioni di aspetto generale che riguardano tutti i file, presenti e futuri, inseriti nei nostri dischi.

Se i file presenti in queste directory non hanno estensione significa che sono file eseguibili e che quindi possono essere lanciati digitando semplicemente il loro nome, preceduto eventualmente dal precorso o path. I file con l'estensione .info vengono utilizzati dal sistema operativo per visualizzare l'icona relativa al programma. Se i file hanno l'estensione .doc oppure .txt significa che siete in presenza di un file di testo, che può contenere eventualmente alcune spiegazioni. Il numero più nutrito di estensioni riguarda i vari file sorgenti dei diversi linguaggi le cui implementazioni sono disponibili per Amiga. Vediamoli.

.bas	sorgente in BASIC
.c	sorgente in C
.h	sorgente header in C
.asm	sorgente in Assembler
.def	sorgente modulo definizione in Modula-2
.mod	sorgente modulo implementazione in Modula-2
.f	sorgente Forth
.scr	sorgente screen Forth
.o	file oggetto non linkato
.obj	file oggetto non linkato
.lsp	sorgente in Lisp

Queste estensioni devono intendersi di riferimento e ogni volta che ci saranno delle variazioni oppure introdurremo nuove estensioni, ne forniremo adeguata documentazione.

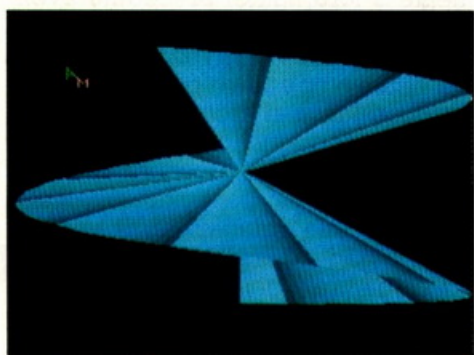
Alcuni programmi è necessario risiedano nell'area di memoria denominata CHIP MEMORY, costituita dai primi 512 K, per il loro corretto funzionamento. Se il vostro sistema è dotato di espansioni che aumentano la quantità complessiva della memoria oltre i 512

K è necessario che, prima di lanciare tali programmi, selezioniate l'icona NOFASTMEM presente nel drawer SYSTEM del Workbench disk.

Witz

Piccole raffinatezze, ammiccanti. Come definire altrimenti un pesce che nuota attraverso il vostro schermo e dei vermiciattoli di colore che inseguono il vostro puntatore?!

TFISH è il nome del programma del pesce che nuota attraverso il vostro schermo. Il programma può essere lanciato da CLI oppure da Workbench, selezionando l'icona relativa.

Per fermare il programma muovete il mouse sulla parte superiore dello schermo, premete il pulsante sinistro del mouse e abbassate lo schermo fino a rendere ben visibile la window di controllo, a questo punto selezionate il box piccolino visibile alla sua destra.

Anche BLOBS può essere lanciato sia da CLI che da Workbench. il titolo del programma non poteva che richiamare le macchie di colore vermiformi che si muovono sullo schermo durante il programma. Premendo il pulsante destro del mouse potete accedere ai menu che sono nell'ordine: Show Menu Bar, che rende visibile la barra del menu; Hide Menu, che la nasconde; Flash, che permette alle macchie vermiformi di lampeggiare cambiando colore; Chase, che ordina ai vermiformi di inseguire il vostro puntatore; Quit, per uscire dal programma.

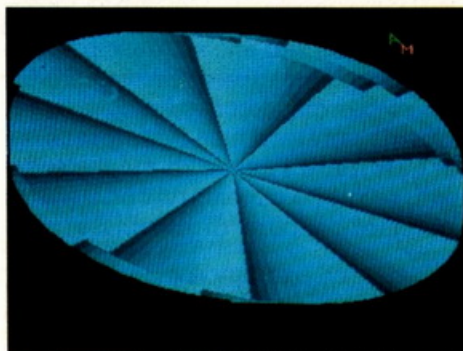
Come detto precedentemente, in questa directory sono presenti i sorgenti in C di entrambi i programmi.

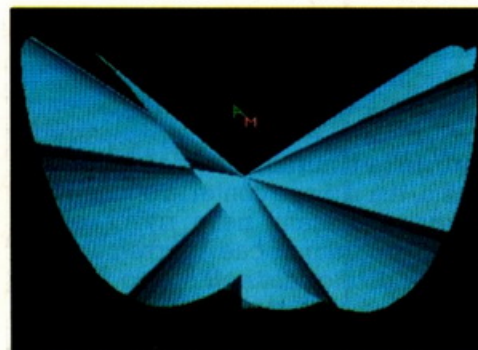
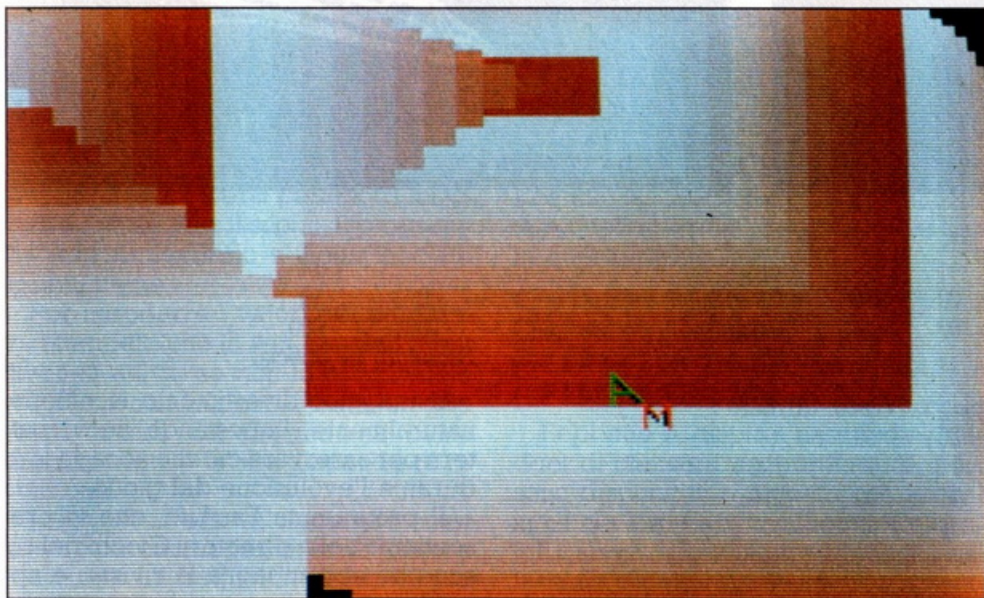
Il ricco menu di Disk Magazine.

Giochi

" Sì, il mio protagonista è svitato, ma non quanto me. Sono io l'idiota che giocò a slot machine contro quel bandito di un computer al Union Carbide e ci rimise anche la camicia".

A. Bester

Noi faremo tutto e di tutto per il vostro divertimento. I programmi contenuti in questa directory o drawer sono appunto dei giochi, Reversi e FastLife rispettivamente, il cui codice sorgente non è presente. Entrambi sono davvero rapidi nell'esecuzione, completi e spettacolari. Entrambi ancora possono venire eseguiti da Workbench oppure da CLI.

Reversi

Reversi è un ottima implementazione multitasking di un gioco inventato alla fine del XIX secolo, e che è ritornato in auge in epoca recente. Il menu non viene visualizzato se non utilizzate il pulsante destro del mouse di Amiga. I menu a vostra disposizione sono Game, Voice, Level e Options. Utilizzate sempre il mouse per effettuare le vostre mosse, muovendo il puntatore sulla casella in cui desiderate spostare la pedina e quindi premendo il pulsante sinistro del mouse. Il movimento viene eseguito quando rilasciate il pulsante. In questo modo avete tutto il

caselle e non si può effettuare nessun'altra mossa. Vince il giocatore che ha il maggior numero di pedine con il suo colore. Si possono fare solo mosse per mangiare e il giocatore che non riesce a fare una presa perde il turno. Questo accade quando voi non avete a disposizione mosse consentite e siete costretti a selezionare la voce Forfeit dal menu Options, dal momento che il programma non compie questa operazione automaticamente. Sempre in questo menu ci sono altre due utilissime voci. Il comando Moves vi presenta infatti tutte le mosse permesse e il comando Suggest vi suggerisce una buona mossa, anche se non la migliore! Riteniamo che l'apprendimento del gioco con l'utilizzo di questi comandi sia semplice e immediato.

tempo per visualizzare sulla tastiera le vostre eventuali mosse. Con l'opzione Voice potete sentire l'Amiga che vi dice, purtroppo in inglese, quando tocca a voi la mossa o fa altri piacevoli commenti. Questo programma viene eseguito a una priorità sufficientemente bassa da permettere, mentre pensa, un buon tempo di risposta se voi lavorate (RUN) con altri programmi interattivi, come per esempio un editor. Un'altra caratteristica importante è che la tastiera viene visualizzata in uno screen e non in una window, consentendovi così di scoprire il Workbench se calate lo screen del programma, oppure di utilizzare il back/front gadget o gadget di profondità.

E veniamo ora alla tastiera, alle pedine, alle mosse e contromosse e al loro significato. Innanzitutto la scacchiera è composta da 64 caselle e sono necessarie 64 pedine, cioè tante quante sono le caselle.

Voi siete il BIANCO.

Dopo aver disposto le prime quattro pedine il computer attende la vostra mossa.

La mossa deve intrappolare una o più pedine nere con la vostra pedina bianca. Il gioco si conclude quando si sono riempite con le pedine tutte le

Si effettua una mossa utile alla presa se si dispongono le pedine in modo da intrappolare perlomeno una pedina avversaria entro una linea, in qualsiasi direzione, diagonalmente, orizzontalmente e verticalmente, tra la prima e un'altra pedina di chi prende. Le pedine prese assumono il colore dell'avversario.

Il livello 5 è il livello più difficile e il programma impiega una maggiore quantità di tempo per muovere le pedine a questo livello. Potete cambiare livello anche durante il gioco. La frase (Voice) "It is your move" non viene pronunciata a livello 1 e 2 , dal mo

mento che il computer impiega più tempo per pronunciarla che per fare la mossa. Le altre divertenti espressioni rimangono. Ricordate infine che quando viene visualizzato il box Busy della mossa permessa, tutti i menu sono disabilitati.

FastLife o il gioco della vita

Il gioco Vita (Life) è stato inventato dal matematico dell'Università di Cambridge John Horton Conway nel 1968. Il gioco si basa su un automa cellulare e si gioca su una griglia bidimensionale. I lettori che desiderano approfondire l'argomento possono consultare l'articolo A.K.Dewdney in Le Scienze n.224, Aprile 1987, (RI-)CREAZIONI AL CALCOLATORE. In questo gioco ci sono solamente due regole. Le regole della Nascita e della Morte.

Nascita: una cellula nasce in qualsiasi quadrato con tre e solamente tre adiacenti vivi o vicini.

Morte: Una cellula muore di isolamento se ha meno di due adiacenti vivi o vicini, e muore di sovraffollamento se ha più di tre adiacenti vivi o vicini.

Gli adiacenti includono adiacenti diagonali, orizzontali e verticali, in questo modo ciascun quadrato ha otto luoghi per i potenziali adiacenti. Attraverso il succedersi nel gioco delle diverse generazioni, le cellule nascono e muoiono con queste regole. Le cellule appena nate possono solamente far nascere oppure aiutare a morire le altre cellule nella generazione successiva alla loro nascita. Come vi cimenterete con diversi pattern o modelli di avvio del gioco, vi diverranno familiari un certo numero di forme stabili ricorrenti. Alcune di queste sono:

```
          0         0
 00      0 0       0 0
 00       0        0 0
                    0
Blocco  Vasca   Alveare
 00      00
 0 0     0 0      0 00
 0 0     00       00 0
  0
Sega  Barcone Serpente
```

Ci sono pure forme oscillanti, cioè quelle che ripetono se stesse più volte:

```
              00 00     000
              00 00     000
    000        0 0      000
              0 0 0 0   000
              0 0 0 0   000
              00 00     000
Lampeggiatore Acrobata Figura 8
      0          0
      00         00
      00         00
      0          0
  Orologio      Faro
```

Ancor più interessanti sono le figure che ripetono se stesse mentre si muovono attraverso lo schermo e che vengono denominate alianti:

```
        0
 0      0
 0     0 0
000   0000

 0      0
 0      0
 0      0
0 0    0 0
0000  0000
```

Ed infine, ci sono figure semplici che possono espandersi simmetricamente oppure asimmetricamente, creando eleganti pattern:

```
            00
   0        0
  000      000
Semaforo  LP Special
```

Il maggior divertimento si ottiene, naturalmente, impostando nuovi pattern per osservare ciò che accade loro durante l'evoluzione del gioco.

Il programma FastLife, che vi presentiamo nella directory Giochi del disco che accompagna la rivista, è appunto una versione rapida e curata del famoso gioco matematico Vita. In questo programma avete a disposizione due menu: quello a sinistra permette di impartire comandi, mentre quello a destra determina le dimensioni delle cellule del gioco. La dimensione più piccola delle cellule vi presenta uno screen davvero ampio, ma risulta difficile disegnare accuratamente in questa modalità. Per questo motivo vi consigliamo l'utilizzo delle prime tre dimensioni di cellule, almeno finché non avete qcquisito una sufficiente maestria.

Nel menu dei comandi potete trovare le voci seguenti: CLEAR SCREEN, che cancella lo schermo, GO che avvia il processo della Vita, STOP che ferma il processo della Vita e QUIT che termina il programma.

Per disegnare le cellule basta premere il pulsante sinistro del mouse; una cellula comparirà nella posizione del cursore del mouse. Se voi muovete il mouse, tenendo premuto il suo pulsante sinistro, verrà disegnata una stringa di cellule in qualsiasi direzione voi abbiate mosso il cursore. Per cancellare le cellule basta che premiate il pulsante sinistro del mouse sulle cellule di cui desiderate sbarazzarvi. Un ultimo avvertimento: per questo come per altri programmi presenti nel disco che vi presentiamo, è necessario che

vengano caricati e lanciati nell'area di memoria denominata CHIP MEMORY e che è costituita dai primi 512 K di memoria.

Musica

Il programma Digital Filter, che vi presentiamo in questa directory, presenta una grafica 640x400 molto curata e richiede in input diversi parametri, - taglio di frequenza, risonanza e forma d'onda -; quindi visualizza l'effetto dei cambiamenti subiti dalla forma d'onda in relazione ai diversi parametri introdotti e ne produce il suono. Il programma è in AmigaBasic e viene lanciato normalmente.

Per introdurre i valori dei parametri, attendete che la window dell'input divenga attiva, se non lo è, attivatela voi selezionandola con il puntatore. Se volete uscire dal programma basta che rispondiate 'n'alla domanda di ripetizione.

Inserite i diversi valori come se essi fossero parametri per un sintetizzatore analogico. La frequenza di taglio, per esempio, potrebbe essere introdotta come un valore percentuale dell'intervallo di frequenze disponibili sul vostro sintetizzatore. E in questo modo regolatevi per gli altri parametri.

Dopo che e' stato visualizzato ciascun grafico viene prodotto il suono alla frequenza fondamentale di 440 Hz.

Quando sono stati visualizzati tutti i grafici voi potete avanzare uno qualsiasi di essi con il gadget di profondità e selezionarlo in qualsiasi suo punto per ascoltare nuovamente il suono relativo.

Questo programma come tutti gli altri programmi in AmigaBASIC è facilmente personalizzabile.

Grafica

Questa directory contiene entrambi i programmi scritti in AmigaBASIC. ColorArt crea una grafica molto bella a base di screen animati dal ciclo di alcuni colori. Ci sono infatti 6 o 7 differenti screen che vengono manipolati e il programmma impiega davvero poco tempo ad attraversarli tutti.

Il programma gira normalmente. Lo si lancia e lo si termina premendo semplicemente il pulsante sinistro del mouse quando più ci garba.

Shuttle genera un'immagine tridimensionale della navetta Shuttle che può essere ruotata e osservata. Anche questo programma non può che girare normalmente. Dopo pochi secondi vi si chiederanno gli angoli di rotazione dell'immagine. Incominciate con 0,0,0 per un migliore risultato. X è l'asse orizzotale, Y l'asse verticale e Z l'asse di profondità. Per fermare il programma premete CTRL-C oppure selezionate STOP dal menu Run.

Strumenti

Innanzitutto mb. Questo programma, eseguibile da CLI oppure da Workbench, visualizza la memoria utilizzata e quella disponibile sulla menu bar in questo ordine:

Chip memory utilizzata
Chip memory disponibile
Fast memory utilizzata
Fast memory disponibile

Sebbene il display nasconda i gadget di chiusura e di profondità, essi sono disponibili e pienamente accessibili nei loro rispettivi luoghi. Utilizzateli dunque senza timore, come se essi fossero visibili, ma senza per questo pensare che l'Amiga sia popolato da strane entità e spiritiche presenze.

Il sorgente in C di questo programma è pure presente in questa directory.

Questa directory contiene anche un'altra directory, Driver_stampanti. Quest'ultima directory contiene driver per otto diverse stampanti non supportate dalla attuale versione di Preferences, più una versione aggiornata del driver Epson che corregge l'errore nella sezione grafica. Eccole:

C_Itoh_Pro - driver per C. Itoh Prowriter

Epson - driver Epson aggiornato

Epson_LQ800 - driver Epson LQ800

ImageWriterII - driver per Apple ImageWriter II

NEC_8025A - driver per NEC 8025A SpinWriter

Okidata-92 - driver per Okidata 92

Panasonic_KX - driver per stampati Panasonic KX-P10XX (1080, 1091, ecc.) Si possono utilizzare i modi superscript e subscript in letter quality, nonostante le indicazioni contrarie del manuale.

Smith-Corona_D300 - driver per Smith-Corona D300

Star_10 - driver per Gemini Star-10

Per utilizzare alcuni di questi driver, innanzitutto dovete introdurvi nel CLI e copiare il driver da voi prescelto nella directory DEVS/PRINTERS del disco Workbench, a questo punto sarà sufficiente attivare il Preferences e per suo mezzo effettuare le modificazioni che ritenete indispensabili. Ecco quali sono i passi che dovete effettuare. Attivate la vostra stampante come Custom, selezionando il box sotto l'area della sezione stampanti ed immettetendo quindi il nome del driver della stampante da voi precedentemente copiato nella directory DEVS/PRINTERS. Salvate questa nuova configurazione ed uscite da Preferences.

Ricordiamo ancora che ai driver della directory Driver_stampanti si può accedere solamente attraverso il CLI.

Magazine

Speriamo intanto che il titolo di questa directory risuoni, come il titolo della rivista, di incontenibili aspirazioni. Con questa directory abbiamo voluto far fronte a diverse prerogative che vogliamo caratterizzino la nostra pubblicazione. Innanzitutto la necessità di assicurare, comunque, la correttezza dei listati, indipendentemente sia da eventuali errori tipografici, che dall'utilizzo di compilatori diversi da quelli utilizzati dall'autore del programma. I più pigri avranno certamente capito che questa directory eviterà loro la fatica di digitare i programmi, consentendo a noi, d'altra parte, di non esitare a presentare anche programmi impegnativi e di una certa mole. Avrete già capito che il contenuto di questa directory sarà costituito da tutti i listati che compaiono nella rivista.

... e al prossimo incontro.

" ... dice che passerà sopra la vostra zona fra cinque minuti circa. Gli piacerebbe che lo salutaste. Dice che la sua orbita non lo porterà a sorvolarvi ancora se non fra un paio di mesi. Quando ripasserà, proverà a telefonarvi direttamente. Arrivederci. " Something up there likes me, A. Bester

" Rido molto, con voi e di me stesso, e la mia risata è sonora e disinibita. Sono un tipo piuttosto chiassoso. Ma non lasciatevi ingannare, anche quando faccio il buffone. La mia mente da gazzaladra è sempre in attesa di poter rubare qualcosa".

A. Bester

Tabelle, ricerca sequenziale, binaria, tecniche Hash

STRUTTURE DATI

di Emilio Orione

Iniziamo una serie di articoli rivolti ai programmatori, a chi tenta di sfruttare tutta la potenza dell'Amiga usando linguaggi più evoluti del vecchio Basic e, per questo, ha bisogno di strumenti di più alto livello. Lo scopo di questi articoli è quello di fornire tutte le tecniche necessarie a una programmazione strutturata, con particolare riguardo all'Amiga.

Inizieremo ad illustrare le strutture dati: cosa sono, come si usano, come si implementano nei vari linguaggi, poiché sono la base della maggior parte dei programmi, per poi passare alla programmazione Top-Down, alle grammatiche, al metodo di Jackson e a molto altro ancora.

Cercheremo quindi di fornire a chi si accinge a programmare con l'Amiga delle tecniche di progettazione di un programma che rendano più facile la sua scrittura, la sua comprensione da parte di terzi, la sua modifica ed il debugging.

Programmare l'Amiga usando linguaggi diversi dal Basic, quali il C, il Pascal, il Modula II o altri, richiede una progettazione del programma più attenta e strutturata, non è più possibile sedersi davanti alla tastiera e iniziare a scrivere, bisogna progettare con cura quello che si vuole ottenere.

Iniziamo quindi il nostro viaggio attraverso l'ingegneria del software, gli argomenti trattati rientrano più o meno in questa materia, parlando di strutture dati.

I dati, questi sconosciuti

La prima domanda che ci poniamo è: cosa è una struttura dati? È un particolare modo di organizzare i dati usati da un programma, caratterizzato dalle operazioni disponibili sui dati stessi.

Ci sono due punti di vista differenti con cui guardare al dato,da un lato il dato può essere visto guardando alla sua struttura interna,cioè come è stato fisicamente implementato nel linguaggio (con array, puntatori...), ed è il modo classico a cui si è abituati,dall'altro possiamo guardare al dato dall' esterno, vedendo la sua interfaccia con il mondo, una visione globale che

tiene conto solo delle operazioni disponibili sul dato stesso.

La visione esterna è usata per progettare un dato ed è tutto quello che occorre conoscere per poi usare quel particolare tipo di dato, l'implementazione poi,può essere fatta in tantissimi modi diversi, ma la struttura dati non deve dipendere da essa.

Dobbiamo cambiare il nostro modo di pensare al dato,che diviene un insieme di procedure, di "primitive", che operano sul dato stesso.

Per fare un esempio il 'tipo' intero e' un dato caratterizzato dalle varie operazioni che si possono compiere su di esso, somma, prodotto etc....,non è quel particolare modo di vedere il numero intero usato dal particolare compilatore.

Nel corso di questi articoli caratterizzeremo le più tipiche strutture dati come lo stack, le tabelle, le liste, i buffer circolari e così via, vedendo le primitive necessarie a gestirli e qualche esempio implementativo sia a livello di primitive, sia come uso nei programmi di queste strutture dati.

Una comodità atta a facilitare la vita del programmatore disponibile con i linguaggi strutturati, è la possibilita di scrivere procedure e funzioni compilabili separatamente dal programma che di fatto servono a aumentare il set di istruzioni del linguaggio stesso.

Sfruttando questa possibilità potremo organizzare le procedure di implementazione dei dati che illustreremo nel corso di questi articoli, in librerie esterne di funzioni da usare quando è necessario; in questo modo avremo creato nuovi "tipi" a disposizione del nostro linguaggio e quindi potremo avere variabili intere, reali, ma anche variabili stack o tabelle. Per esempio il C non dispone di istruzioni che manipolano le stringhe, ma di una serie di funzioni di libreria standard, cioè disponibili in tutti i compilatori, che svolgono le normali operazioni sulle stringhe permesse dal Basic o dal Pascal con semplici assegnazioni, tipo strcpy(s1,s2) che copia s2 in s1, che in Pascal sarebbe s1:=s2.

A questo punto creando una nuova serie di funzioni che implementino le operazioni permesse su uno stack, tipo pop(), push(), top(), avremo a disposizione, nel C come in altri linguaggi di un nuovo tipo, lo stack, usabile in modo analogo alle stringhe.

Le tabelle

Finito il discorso generale passiamo all'argomento di questo mese, le Tabelle.

Vedremo innanzitutto la definizione di tabella, poi i vari tipi di ricerche che si possono effettuare su di essa, con degli esempi di implementazione e infine un esempio di uso di una hash table in un programma vero e proprio scritto in C, chiamato TkCr (sta per Token Corrector), che serve ad evidenziare le parole chiave di un linguaggio rendendole maiuscole, un po' come fa l'editor dell'AmigaBasic.

Si definisce "Tabella" un gruppo di elementi detti "Record", ad ognuno dei quali è associata una "Chiave" che è usata per differenziare record diversi.

Le chiavi poi possono essere di due tipi: semplice o esterna.

Una chiave è detta semplice, in inglese embedded key (chiave incastrata), quando fa parte del record stesso, è un campo del record, per esempio è una tabella a chiave semplice un array che ha come elementi dei record e come chiave l'indice stesso dell'array.

```
| Chiave | Record |
|        |        |
|        |        |
|        |        |
|        |        |
|        |        |
|        |        |
```

Chiave interna.

Una chiave è detta esterna quando l' insieme delle chiavi risiede in una tabella separata con puntatori alla tabella dei record.

```
/* >>>>>>>>>>>>>>>>>>>>>>>>>> Token Corrector
<<<<<<<<<<<<<<<<<<<<<<<<<< *
 *
  *
 * Corregge un file Modula II sostituendo alle parole con le maiuscole
  *
 * sbagliate le parole corrette.
  *
 *
  *
 * Il file "Token", contenente le parole chiave del Modula II scritte
  *
 * correttamente, deve essere presente nella directory corrente.
  *
 *
  *
 *                                     ..... 1987 by Emilio Orione.
  *
 *
  *
 * >>>>>>>>>>>>>>>>>>>>>>>>>> Token Corrector
<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include "stdio.h"
#include "ctype.h"

#define MAXL 255     /* Lunghezza massima dell'input */
#define HTSIZE 127  /* Ampiezza della hashtable,deve essere un numero
primo
                     */
#define VOID int
#define BOOL int
#define TRUE 1
#define FALSE 0
#define EQUALS 0

char  hashtable[ HTSIZE ][ MAXL ] ;

VOID stripnewlines (stringtoprocess) /* Elimina '\n' dalla stringa */
char *stringtoprocess ;
{
   while (*stringtoprocess && *stringtoprocess != '\n')
       stringtoprocess++ ;
     *stringtoprocess = '\0' ;
}

VOID lowerstring (value) /* Rende minuscola una stringa */
char *value ;
{
   for ( ; *value != '\0' ; value++)
      *value = tolower(*value) ;
```

Si possono fare ulteriori suddivisioni sulle chiavi: si parla di chiave primaria se la chiave identifica univocamente un record (come per l'indice di un array) o di chiave secondaria se la chiave identifica più record.

Prendiamo per esempio una tabella usata per contenere le informazioni sui libri di una biblioteca, i suoi record avranno una struttura di questo tipo (usando per descriverla un linguaggio algoritmico simile al Pascal):

```
Type TipodelRecord = Record
                        Autore:  string[20] ;
                        Anno:    string[4] ;
                        Editore: string[20] ;
                        Titolo:  string[30]
                        ;
                     End ;
```

Nota: le parole sottolineate sono parole chiave del linguaggio,sono simboli terminali.

Se usiamo come chiave il campo autore, avremo definito una chiave secondaria interna poiché vi potranno essere più libri per uno stesso autore, viceversa se usiamo come chiave l'indice di un array di TipodelRecord, cioè facciamo le seguenti dichiarazioni:

```
Const   DimTabella = 100;
Type    Chiave = 1 .. DimTabella ;
        TipodelRecord = Record
                           Autore: string[20] ;
                           Anno: string[4] ;
                           Editore: string[20] ;
                           Titolo: string[30] ;
                           End;
Var     Tab : array[Chiave] of TipodelRecord;
```

otteniamo una variabile di tipo tabella (Tab), che usa una chiave primaria interna, poiché ogni chiave identifica in modo univoco un solo record.

Chiavi multiple

Ci possono poi anche essere tabelle a chiavi multiple in cui l'identificazione di un record avviene attraverso più chiavi. Con questo stesso esempio potremmo utilizzare l'indice dell'array come chiave primaria che identifichi un solo record, i campi Autore e Anno come chiavi secondarie per fare ricerche che producano un insieme omogeneo di record.

Come abbiamo detto il particolare modo

```
}

int transformid( word)
char word[] ;
/*
   Converte una stringa in un intero che,modulo HTSIZE,fornisce
l'indice
   perla tabella hash.
*/
{
   int term ;
   int wordindex ;

   for (term = 0 , wordindex = strlen(word) -1 ; wordindex > -1 ;
wordindex--)
       term = (257*term) + word[wordindex] ;
   term = (term < 0) ? -term : term ;
   return( term % HTSIZE) ;
}
int generatenewindex( originalkey,probenumber)

int originalkey ;
int probenumber ;
/*
   Genera un nuovo indice per la tabella hash in base all'indice
precedente
   per risolvere una collisione.Viene usato il quadrato dell'indice
   precedente per riempire la tabella in modo piu' uniforme.
*/
{
   return( (originalkey + probenumber * probenumber) % HTSIZE) ;
}
int hash(word)
char *word ;
/*
   Retistuisce un indice per word sicuramente libero da collisioni.
   Se la tabella e' piena restituisce -1.
*/
{
   int htindex ;
   int idlen ;
   int inithtindex ;
   int probecounter = 0 ;

   if (!(idlen = strlen(word)))
       printf( "Hash : Word of no lenght\n") ;

   /* Calcolo dell'indice iniziale per word */

   htindex = inithtindex = transformid( word) ;

   /* Se la tabella e' vuota siamo a posto */

   if (*hashtable[ htindex ] == '\0')
       ;
   else
   /* Una collisione si cerca un altro indice */

       for (probecounter = 0 ; probecounter < (HTSIZE / 2) ;
                   probecounter++) {
           htindex = generatenewindex( inithtindex,probecounter) ;
           if (*hashtable[htindex] == '\0') /* Trovato */
                   break ;
       }

   /* Sono stati fatti troppi tentativi,la tabella e' piena.
      L'errore e' segnalato con -1
   */

   if (probecounter >= (HTSIZE /. 2))
```

di implementare una tabella non deve influenzare l'uso della tabella stessa, occorrono cioè delle procedure che racchiudano al loro interno tutto quello che è implementazione della tabella, e che permettano al mondo esterno di operare sulla tabella.

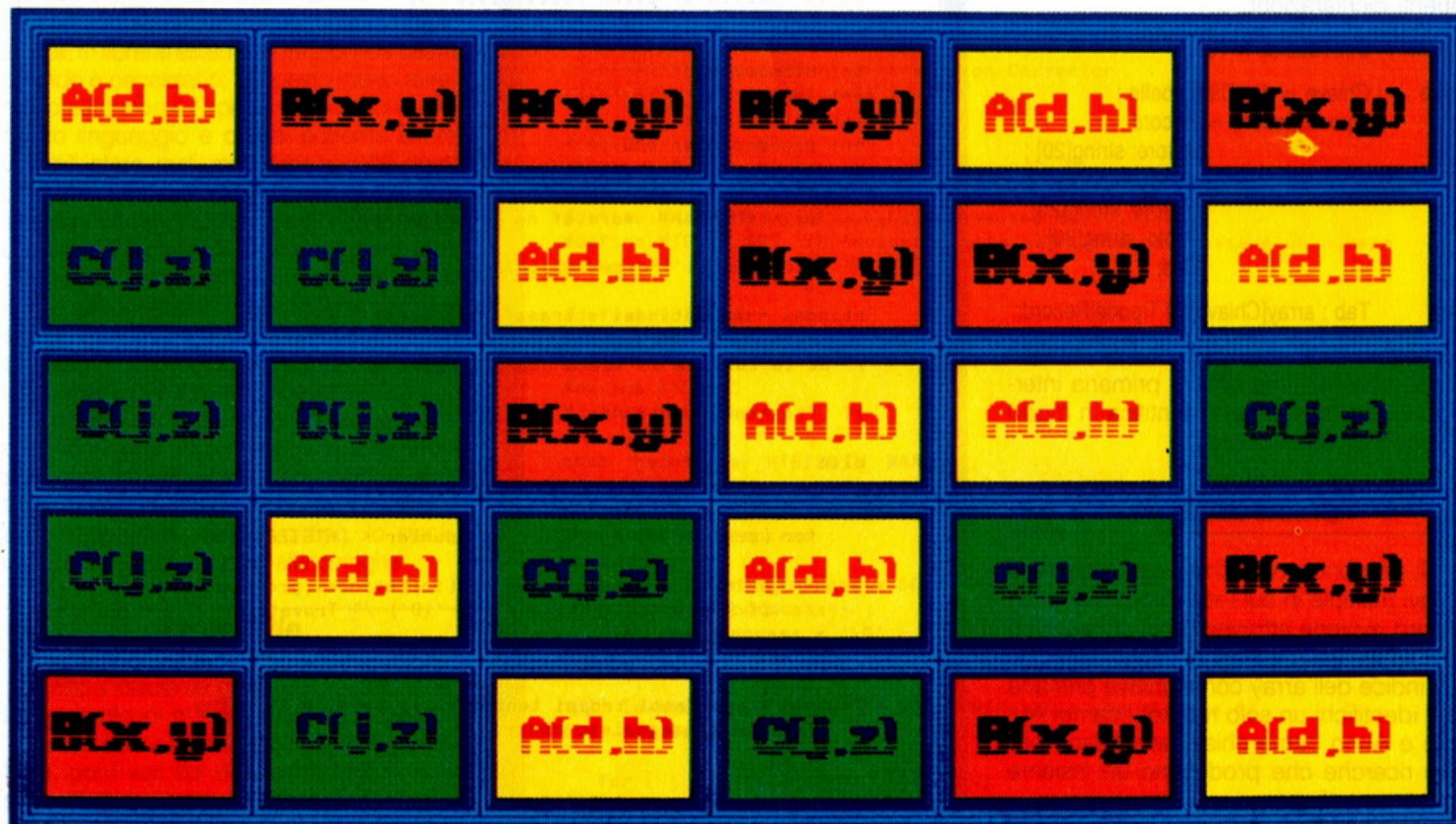
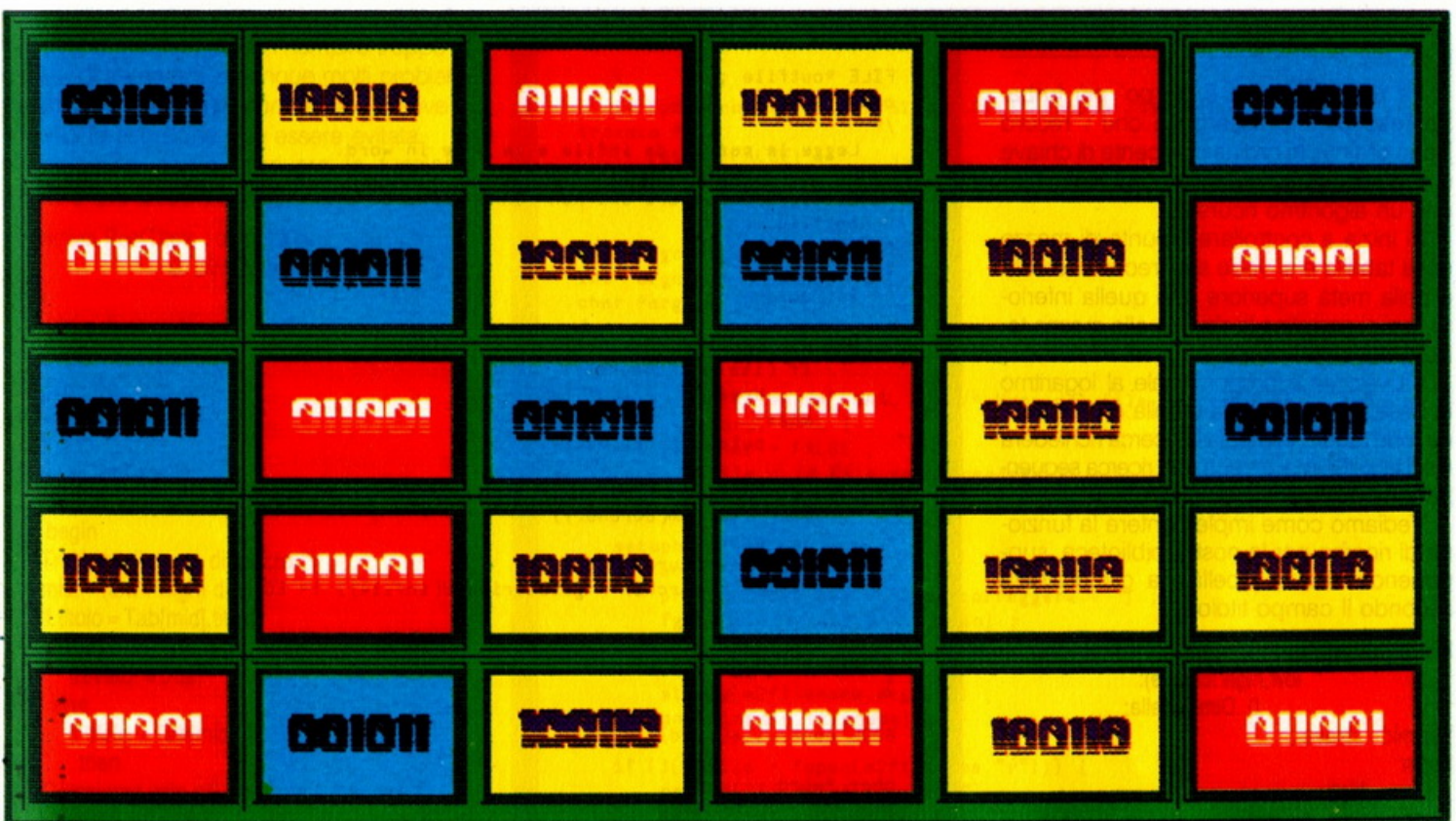
In pratica le operazioni standard che si devono compiere in una tabella sono l'inserimento e l'accesso ad un dato elemento cioè la ricerca. Per fare un inserimento però, prima bisogna cercare un posto vuoto, poi inserirvi il record, quindi la sua velocità dipende dall'algoritmo di ricerca.Questo può essere fatto in modi diversi e influenza pesantemente le prestazioni di un programma.

Ovviamente dovrà esserci un valore della chiave che indichi che il record è vuoto per poter usare la stessa procedura di ricerca, sia per inserire sia per cercare un determinato record.

La ricerca

Ci sono tre modi di effettuare una ricerca: si può fare una ricerca sequenziale, una ricerca binaria, usare una tabella hash.

Il modo con cui si decide di effettuare la ricerca non deve influenzare il resto del programma; si crea una procedura a cui vanno passate sia tabella sia la chiave che ritorna il record corrispondente, il modo con cui la procedura lavora non deve dipendere dal resto del programma.

```
        return(-1) ;
    return(htindex) ;
}

BOOL insertword(indx,word)
char *word ;
char *indx ;
/*
    Inserisce una nuova parola nella tabella hash.
    Ritorna TRUE se tutto e' a posto,FALSE se la tabella e' piena.
*/
{
    int htindex ;

    if ((htindex = hash(indx)) == -1)
        return(FALSE) ;
    if (*hashtable[htindex] == '\0')
        strcpy(hashtable[htindex],word) ;
    return(TRUE) ;
}

VOID setupht()
/*
    La funzione inserisce i Token (parole corrette) nella tabella hash.
    Si suppone che il numero di parole sia il 70% di HTSIZE.
    Le parole sono lette dal file Token preso nella directory corrente.
*/
{
    int   counter ;
    FILE  *infile ;
    char  instring[ MAXL ] ;
    char  indice[ MAXL ] ;

    for (counter = 0 ; counter < HTSIZE ; counter++)
        hashtable[counter][0] = '\0' ; /* Inizializza tabella */
```

```
    if ( !( infile = fopen( "Token","r"))) { /* Apre il file Token */
         printf( "Dov' e' il file 'Token'\n") ;
         exit(0) ;
    }

    while ( fgets(instring,MAXL,infile) ) {
         stripnewlines( instring) ;
         strcpy( indice,instring) ;
         lowerstring( indice) ;
         if ( !(insertword( indice,instring))) {
              printf( "Hash Table Full!\n") ;
              exit(0) ;
         }
    }
    fclose( infile) ;
}

int getcstripped( whichfile)

FILE *whichfile ;
/*
    Restituisce un carattere letto da whichfile eliminando (se non si
tratta
    di EOF) l'ottavo bit.
*/
{
    int temp ;

    if ((temp = getc(whichfile)) != EOF)
         return( temp & 0x7f) ;
    return(EOF) ;
}

int getword( infile,outfile,word)

FILE *infile ;
```

Potremmo decidere per semplicità di effettuare una ricerca sequenziale, poi per ragioni di efficenza passare ad una ricerca binaria, riscrivendo solo la procedura di ricerca e non tutto il programma. Una ricerca sequenziale controlla tutta la tabella partendo dall'inizio fino a che non trova il record.

Con le stesse dichiarazioni di prima, una funzione che, passatagli come chiave un titolo, ritorni la prima posizione in tabella del record che lo contiene, è la seguente:

```
Function Pos(Titolo : string[30]):
                    0..DimTabella ;
Var Trovato : boolean ;
     i : integer ;
begin
   found: = false ; i: = 1 ;
   while (i < = DimTabella AND
                   (NOT trovato)) do
     if titolo = Tab[i].titolo
       then
(* Se sono uguali l'abbiamo trovato *)
         begin
            Pos: = i ;
            trovato: = true
         end ;
(* Altrimenti continuiamo con il prossimo record *)
     else
          i: = i + 1 ;
   if NOT trovato then Pos: = 0
(* Zero indica l'insuccesso *)
end ;
```

La ricerca binaria

La ricerca binaria è un po' più furba. Per eseguirla è essenziale che i record siano ordinati in ordine crescente di chiave e si presta molto ad essere implementata con un algoritmo ricursivo.

Si inizia a controllare il punto di mezzo della tabella e si vede se il record cercato è nella metà superiore o in quella inferiore, poi si riapplica la ricerca alla mezza tabella così individuata.

La velocità è proporzionale al logaritmo in base due, così se la tabella è composta da mille record in media la ricerca richiederà dieci tentativi, mentre con una ricerca sequenziale i tentativi sarebbero stati cinquecento.

Vediamo come implementere la funzione di ricerca per la nostra biblioteca, supponendo che la tabella sia già ordinata secondo il campo titolo:

```
Function Pos(Titolo : string[30],
                low,high :chiave):
                    0..DimTabella;
Var mid : chiave ;
begin
  if low > high
   then
(* Non c'è il record che cerchiamo Zero indica
   l'insuccesso *)
    Pos: = 0
  else
    begin
(* Cerchiamo il punto di mezzo *)
     mid: = (low + high) div 2;
     if titolo = Tab[mid].titolo
       then
       (* Trovato! *)
         Pos: = mid
       else
         if titolo < Tab[mid].titolo
          then
(* Continuo ricursivamente la ricerca nella metà
          inferiore *)
          Pos: = Pos(titolo,low,mid-1)
         else
(* Allora è nella metà superiore *)
       Pos: = Pos(titolo,mid + 1,high)
    end
end;
```

Per chiamare la funzione dal resto del programma dovrò passargli il limite inferiore e quello superiore di ricerca, la chiamata sarà fatta quindi in questo modo:

```
Posizione: = Pos(xxxxxx,1,DimTabella);
```

dove posizione è una variabile di tipo chiave e "xxxxxxx" il titolo cercato.

L'algoritmo appena descritto è ricursivo,cioè la funzione chiama più volte se stessa per trovare la soluzione.

Il problema è che alcuni linguaggi non

```
FILE *outfile ;
char *word ;
/*
   Legge le parole da infile e le pone in word.
   Se incontra EOF lo ritorna.
   Se legge caratteri che non fanno parte di una parola li riscrive in
   outfile.
*/
{
   int curchar ;

   while ((curchar = getcstripped(infile)) != EOF)
       if (isalpha(curchar))
           break ;
       else
           putc(curchar,outfile) ;

   do {
       if (!isalnum(curchar))
           break ;
       *word++ = curchar ;
     } while ((curchar = getcstripped(infile)) != EOF) ;

   *word = '\0' ;
   return (curchar) ;
}

BOOL controlla(word)

char *word ;
/*
   Controlla se word e' presente nella hashtable e nel caso la
sostituisce
   con la versione corretta.
   L'indice viene sempre generato con la parola tutta minuscola per
evitare
   che un carattere maiuscolo falsi il riconiscemento.
   Se vi sono troppe collisioni e la parola non si trova ritorna
   FALSE,viceversa ritorna TRUE.
*/
{
   int htindex ;
   int inithtindex ;
   int probecounter = 0 ;
   char indice[ MAXL ] ;
   char flag[ MAXL ] ;

   strcpy( indice,word) ;
   lowerstring( indice) ;
   htindex = inithtindex = transformid( indice) ;
   if (*hashtable[htindex] != '\0') { /* La parola puo'essere presente
*/
       strcpy( flag,hashtable[ htindex]) ;
       lowerstring( flag) ;
       if (strcmp( flag,indice) == EQUALS) /* L'abbiamo trovata */
           strcpy( word,hashtable[ htindex]) ;
       else
           for (probecounter = 0 ; probecounter < (HTSIZE / 2) ;
                   probecounter++) {
           htindex = generatenewindex( inithtindex,probecounter) ;
           if (*hashtable[ htindex] == '\0')
               break ; /* Non e' presente */
           else {
               strcpy( flag,hashtable[ htindex]) ;
               lowerstring( flag) ;
               if (strcmp( flag,indice) == EQUALS) /* Trovata */
                   {

                       strcpy( word,hashtable[ htindex]) ;
                       break ;
                   }
           }
```

permettono la ricursione ed inoltre certe volte è meglio evitarla per non riempire troppo lo stack di sistema. Comunque molti problemi hanno soluzioni sia ricursive sia iterative e quindi la ricursione può essere evitata.

Vediamo come riscrivere la funzione Pos in modo iterativo:

```
Function Pos(Titolo : string[30]):
                0..DimTabella ;
Var
  mid,low,high : chiave ;
  trovato : boolean ;
begin
trovato: = false;
low: = 1;
high: = DimTabella;
while (low  < high) AND (NOT trovato)
do begin
(* Troviamo il punto di mezzo *)
   mid: = (low + high) div 2;
   if titolo = Tab[mid].titolo
     then
       trovato: = true;
   else
     if titolo  < Tab[mid].titolo
       then
(* Cerchiamo solo più nella metà inferiore *)
         high: = mid-1;
       else
(* Altrimenti è nella metà superiore della tabella  *)
         low: = mid + 1
end;
if trovato then Pos: = mid
            else Pos: = 0
(* Zero indica il fallimento della ricerca *)
end;
```

Le tabelle Hash

Passiamo ora a vedere cosa sono le tabelle Hash.

Il funzionamento ideale, usando le chiavi per fare ricerche in una tabella, sarebbe di trovare subito, al primo tentativo, il record voluto.

Se come chiave usassimo l'indice di un array questo sarebbe possibile, a patto di conoscere già la posizione giusta.

Bisogna trovare un meccanismo che ci consenta di far corrispondere ad un valore della chiave un indice dell'array. Per farlo si determina una funzione di trasformazione hash(C) che "mappi" l'insieme delle chiavi nell'insieme di indirizzi della tabella. Di solito l'insieme [C] è più grande dell'insieme degli indirizzi [I], si dice che la cardinalità di [C] è maggiore di quella di [I],quindi è quasi impossibile trovare una funzione hash che associ un Indirizzo diverso ad ogni Chiave.

Non è comunque un problema gestire un

```
        }
    }
    if (probecounter >= (HTSIZE / 2))/* Troppe collisioni parola non
trovata */
       return(FALSE) ;
    return(TRUE) ;
}

main(argc,argv)
int argc ;
char *argv[] ;
{
   char m2filename[ MAXL ] ; /* Nome del file ModulaII da correggere
*/
   char m2outfilename[ MAXL ] ; /* Nome del file corretto */
   char curword[ MAXL ] ;
   int curchar = FALSE ;
   FILE *m2file ; /* File da correggere */
   FILE *goodm2 ; /* File corretto */

   setupht() ;
   if (argc < 2) {
        printf("Nome del file da correggere ? ") ;
        fgets(m2filename,MAXL,stdin) ;
   }
   else
        strcpy(m2filename,argv[1]) ;
   stripnewlines(m2filename) ;

   if (!(m2file = fopen(m2filename,"r"))) {
        printf("File not found.\n") ;
        exit(0) ;
   }
    strcpy(m2outfilename,m2filename) ;
 /*
   Modificare la linea seguente se non si desidera che al file
corretto sia
   aggiunto il suffisso ".mod".Ricordare pero' che un suffisso e'
necessario
   per differenziare il nome del programma da correggere da quello
corretto
*/
   strcat(m2outfilename,".mod") ; /* Nome del file corretto sara'
                                             "m2filename.mod" */

   if (!(goodm2 = fopen(m2outfilename,"w"))) {
        printf("Can't open %s for output.\n",m2outfilename) ;
        exit(0) ;
   }
   printf("Scrivo il file %s\n",m2outfilename) ;

   while (curchar != EOF) {
        curchar = getword(m2file,goodm2,curword) ;
            if (curchar != EOF) {
                 if (!controlla(curword)) {
                      printf("Collisione nella Hash Table per la parola,
%s\n",
curword) ;
                      printf("Controllo non effettuato su %s\n",curword)
;
                 }
                 fputs(curword,goodm2) ;
                 putc(curchar,goodm2) ;
        }
        strcpy(curword,"") ;
   }
   fclose(m2file) ;
   fclose(goodm2) ;
   printf("Fatto!\n") ;
}
```

conflitto di indirizzo: basta definire un funzione di rehash che gestisca le collisioni. Poniamo per esempio di applicare la funzione di hash ad una chiave k4 e di ottenere l'indirizzo h0 nella seguente tabella:

Indirizzo	Chiave contenuta	Faccio
		hash(k4) = h0 = 2
0		
1		
--> 2	k1	Collisione, la posizione è già occupata
3		Faccio rehash(h0) = h1 = 5
4		
--> 5	k2	Collisione, la posizione è già occupata
6		Faccio rehash(h1) = h2 = 9
7		
8	k3	
--> 9		Perfetto inserisco qui k4

Il "trucco" è di applicare la funzione di hash alla chiave originale e la funzione di rehash all'indice in cui si è effettuata la collisione. Il rehash è fatto, in caso di collisione, sia per un inserimento in tabella sia per una ricerca,se la tabella è fatta bene trova la posizione voluta in pochissimi colpi.

Di solito una buona funzione di hash è: H(K) = ord(K) mod N , dove N è uguale alla dimensione massima della tabella più uno, N = DimTabella + 1 seguendo le dichiarazioni precedenti; ord(k) associa un numero alla nostra chiave, potrebbe essere il codice ASCII del primo carattere della chiave.

L'insieme dei valori prodotti da tale funzione copre tutte le posizioni della tabella, ma non assicura l'assenza di collisioni.

Una semplice funzione di rehash consiste nel provare con la posizione successiva: h1 = RH(h0) = (h0 + 1) mod N , che in generale all'iesimo tentativo diventa: hi = RH(hi-1) = (h0 + i) mod N.

Notare che "h0" è sempre il primo indice trovato con la funzione hash, mentre "i" è il numero di rehash effettuati.

Lo svantaggio di questa funzione è di creare dei grossi blocchi pieni nella tabella, mentre la tecnica hash funziona meglio se la tabella è piena di buchi.

Un'altra funzione di rehash che evita la formazioni dei blocchi nella tabella è il quadrato: hi = RH(hi-1) = (h0 + i*i) mod N.

Questa funzione di rehash è già abbastanza buona; un altro metodo potrebbe essere quello di usare un generatore di numeri pseudocasuale, che migliorerebbe le prestazioni rispetto al metodo quadratico.

L'importante è di non fare blocchi contigui, le chiavi vanno "mappate" nel modo più uniforme possibile, ee inoltre fare in modo che le funzioni possano riempire tutta la tabella senza tralasciare alcuna posizione.

La scelta delle funzioni di hash e di rehash è spesso critica.Da loro dipende l'efficenza della tabella; è utile che l'algoritmo di ricerca e quello di inserimento non dipendano da esse in modo da poterle variare senza problemi.

Un tipico algoritmo di inserimento sarà di continuare a applicare la funzione di rehash fino a che non si trovi un posto libero, però, se la tabella è troppo piena la ricerca del posto vuoto non si fermerà più. Occorre controllare che la tabella non sia piena più dell'80%, oppure fissare un numero massimo di tentativi di rehash prima di uscire con condizione di posto non trovato. Per evitare che, in un programma, si verifichino queste condizioni di errore per tabelle troppo piene, occorre dimensionare accuratamente la tabella in modo che sia in grado di contenere tutti i dati necessari senza riempirsi troppo.

Ottimizzazione di un Hash table

Un modo per migliorare le prestazioni di una hash table è quello di fissare le dimensioni massime della tabella usando un numero primo. Naturalmente più le dimensioni della tabella sono grandi rispetto alla quantità dei dati che essa deve contenere, più le prestazioni saranno migliori,ma ciò porterà ad un maggiore spreco di memoria; la scelta di cosa sia meglio va fatta di volta in volta.

Vediamo una tabella comparativa delle prestazioni in termini di velocità di una hash table usando le due diverse funzioni di rehash viste: indichiamo con A la percentuale di riempimento della hash table, con RL la funzione di rahash lineare e con RQ la funzione di rehash quadratica. Nella tabella viene riportato il numero medio di accessi all hash table per trovare un posto vuoto.

A	RL	RQ
0,1	1,06	1,05
0,5	1,50	1,44
0,75	2,50	1,99
0,9	5,50	2,79
1	Infiniti	Infiniti

Vediamo che anche con la RQ in condizioni estreme, tabella piena al 90%, non si superano i tre tentativi (2,79) per trovare un posto vuoto. In generale la RQ è sempre da preferire, a meno che non si disponga di un generatore pseudorandom. Le hash table sono molto usate, specie per applicazioni in cui la velocità di ricerca è critica, per esem-

pio la symble table di un compilatore è realizzata con tecniche hash.

Uso dell'Hash table

Vediamo un esempio di uso pratico di una hash table. Pur con tutti i suoi difetti l'editor dell'AmigaBasic ha un pregio: evidenzia in maiuscolo le parole chiave del linguaggio abbellendo il programma e rendendolo più leggibile.

A volte scrivere in maiuscolo le parole chiave diventa un fatto non soltanto estetico, ma essenziale al funzionamento del programma. Il compilatore ModulaII della TDI Software Inc. disponibile per l'Amiga distingue fra caratteri maiuscoli e minuscoli e quindi non riconosce le parole chiave non scritte in maiuscolo.

Il poter scrivere tutto un programma senza dover ricorrere sempre allo SHIFT, come avviene per l'AmigaBasic, è senz'altro una comodità ed il programma TkCr serve appunto a fare questo.

Usando TkCr potrete scrivere un programma in ModulaII usando un qualsiasi editor di testi, senza preoccuparvi delle maiuscole, poi invocare TkCr passandogli come argomento il file testo così scritto e vi troverete un nuovo file con l'aggiunta del suffisso ".mod" pronto per essere compilato, con tutte le lettere maiuscole al posto giusto.

Per funzionare TkCr richiede la presenza nella directory corrente di un file chiamato Token contenente tutte le parole da rendere maiuscole scritte in modo corretto.

Queste parole all'atto dell'esecuzione del programma verranno messe in memoria in una hash table e poi confrontate con tutte le parole presenti nel file testo esaminato.

In caso di uguaglianza, (i confronti vengono effettuati a lettere tutte minuscole) la parola contenuta nel file token viene sostituita a quella contenuta nel file di testo.

Per un grosso file sorgente gli accessi alla hash table sono molto numerosi, ma il tempo di esecuzione si mantiene ugualmente molto basso. Lavorando in ram: un file di circa 30K viene corretto in circa tre o quattro secondi, mentre il tempo richiesto da una tabella ad accesso sequenziale o anche binario sarebbe stato proibitivo.

Per chi proprio vuole evitare l'uso dello SHIFT è anche possibile inserire nel file Token i nomi delle funzioni di libreria che verranno corrette come le parole chiave.

Ad esempio il compilatore della TDI vuole che la funzione WriteString sia scritta proprio così, e non writestring o WRITESTRING, basterà quindi inserire nel file Token "WriteString" per doversi più preoccupare di come scriverla.

L'uso di TkCr non è utile solo agli utilizzatori del compilatore ModulaII della TDI, ma anche ad utilizzatori di altri linguaggi. Per esempio l'editor del compilatore True Basic della True Basic inc. non converte le parole chiave del linguaggio in maiuscolo, pur non facendo distinzione fra miuscolo e minuscolo; basta sostituire il file Token presentato in questa rivista con uno che contenga le parole chiave del True Basic, per poter migliorare la leggibilità di tutti i programmi scritti in questo linguaggio.

TkCr

Passiamo ora velocemente ad esaminare più da vicino il funzionamento di TkCr, facendo notare alcune cose non riportate nel commenti. All 'inizio viene dichiarata la costante HTSIZE, che contiene la dimensione massima della hash table.Il valore fissato in 127 (numero primo) e' più che sufficente per il file Token usato.Chi volesse aumentare le dimensioni del suddetto file e si trovasse ad avere una tabella troppo piena, si ricordi di

```
/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Primi
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *
 *
   *
 *  Il crivello di Eratostene permette di calcolare tutti i numeri
primi   *
 *  compresi tra 1 e 'n'.
   *
 *  Uso: Primi -o outputfile [ n ] , con n intero positivo maggiore di
2   *
 *  e minore di 3000.
   *
 *  Viene creato il file "Primi1-n" che contiene i numeri primi.
   *
 *  Se outputfile non e' specificato "Primi1-n" e' creato in ram:.
   *
 *  Se n non e' specificato viene richiesto.
   *
 *
   *
 * >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Primi
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< */

#include <stdio.h>
#define  MAXL 255   /* Lunghezza massima dell'input */
#define  MAXI 3000  /* Limite superiore del crivello */

main(argc,argv)
int argc;
char *argv[];
{
   char outname[MAXL],supascii[MAXL],*s;
   FILE *fopen(),*outfile;
   int out = 0,err = 0;
   long sup,i,j;
   int  crivello[MAXI];

   while (--argc > 0 && (*++argv)[0] == '-') /* Verifica le opzioni */
       for (s = argv[0]+1; *s != '\0'; s++)
           switch (*s) {
               case 'o':
                   out = 1;
                   break;
               default:
                   printf("Primi: opzione scorretta %c\n",*s);
                   err = 1;
                   break;
           }
   if (err == 1) {
       printf("Uso: Primi -o outputfile [ n ]\n");
       exit(0);
   }
   if (out == 1) {
       --argc;
       strcpy( outname,*argv++);
```

settare HTSIZE con un altro numero primo. Siccome calcolare a mente i numeri primi dopo 127 può essere un tantino difficoltoso, abbiamo inserito anche un altro programmino, anche esso scritto in C, che, sfruttando il crivello di Eratostene, calcola tutti i numeri primi compresi fra uno e un numero scelto dall'utente. Continuando il nostro esame di TkCr troviamo la funzione Transformid() che altro non è che la funzione di Hash.Al posto di ord() viene usato un algoritmo un po' più complicato che sfrutta tutti i caratteri della chiave per trasformare la chiave in un numero, anche se questo non cambia di molto le prestazioni.

Subito dopo c'è la funzione GenerateNewIndex() che è una funzione di ReHash quadratico.La funzione nel programma chiamata Hash() è in realtà la funzione di ricerca di un posto vuoto nella tabella.Il numero massimo di tentativi prima del quale fermarsi dichiarando un errore per tabella piena è fissato in HTSIZE / 2, che è sicuramente un numero alto di prove (ricordate che con la tabella piena al 90% il numero medio di tentativi è tre).Per un buon funzionamento si suppone che il numero di parole corrette inserite in Token sia il 70% di HTSIZE, cioè essendo HTSIZE = 127 il numero massimo di parole inseribili in Token è circa 89.

Queste tre funzioni sono di carattere assolutamente generale e riusabili in qualunque programma che usi una hash table.

Il nome del file da correggere può essere inserito nella linea di comando subito dopo il nome del programma, in caso contrario verrà richiesto esplicitamente dal programma.

Ecco due invocazioni corrette di TkCr per correggere il file "testo" :

1) TkCr testo
2) TkCr

Nel secondo caso alla richiesta "Nome del file da correggere ?" si dovrà rispondere "testo". In entrambi i casi verrà prodotto un file "testo.mod" contenente le maiuscole corrette.

Arrivederci

Questo è tutto per il momento, speriamo che gli argomenti trattati questo mese Vi abbiano fornito degli spunti per i vostri programmi, in ogni caso chi volesse presentare delle procedure particolarmente efficenti che implementano delle strutture dati o desiderasse chiarimenti su argomenti particolari può scrivere presso la redazione di "Amiga Magazine". Nel frattempo provate a sperimentare l'uso delle tabelle viste nei vostri programmi e, quello che è più importante per imparare a programmare bene, Compilate gente Compilate !!!

```
    }
    if (argc == 0) {
        printf("Inserisci il limite superiore del crivello: ");
        scanf("%ld",&sup);
    }
    else
        sup = atoi(*argv);
    if (sup < 3 || sup > 3000) {
        printf("Estremo superiore non valido. Range(3-3000)\n");
        exit(0);
    }
    if (out == 0) {
        strcpy(outname,"ram:Primi1-");
        itoa(sup,supascii);
        strcat(outname,supascii);
    }
    if (!(outfile = fopen(outname,"w"))) {
        printf("Non posso aprire %s\n",outname);
        exit(0);
    }
    fprintf(outfile,"1\n");
    for (i = 0; i <= sup ; i++)
        crivello[i] = 0;
    for (i = 2; i <= sup; i++)
        if (crivello[i] == 0) {
            fprintf(outfile,"%ld\n",i);
            for (j = i; j <= sup;  j += i)
                crivello[j] = 1;
        }
    fclose(outfile);
}
itoa(n,s) /* Converte un intero in una stringa */
long n;
char s[];
{
    int i,segno;
    if ((segno = n) < 0)
        n = -n;
    i = 0;
    do {
        s[i++] = n % 10 + '0';
    } while ((n /= 10) > 0);
    if (segno < 0)
        s[i++] = '-';
    s[i] = '\0';
    reverse(s);
}
reverse(s) /* Inverte una stringa */
char s[];
{
    int c,i,j;
    for (i = 0,j = strlen(s)-1;i < j;i++,j--) {
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}
atoi(s)  /* Converte una stringa in un intero */
char s[];
{
    int i,n,segno;
    for (i = 0;s[i] == ' ' || s[i] == '\n' || s[i] == '\t';i++)
        ;
    segno = 1;
    if (s[i] == '+' || s[i] == '-')
        segno = (s[i++] = '+') ? 1 : -1;
    for (n = 0;s[i] >= '0' && s[i] <= '9'; i++)
        n = 10 * n + s[i] - '0';
    return (segno * n);
}
```

CORSO DI A...

di Paolo Russo

Prima puntata di un completo corso di programmazione sul linguaggio Assembly MC68000, il linguaggio macchina dell'Amiga

Il numero di linguaggi di programmazione attualmente esistenti è semplicemente enorme; la maggior parte di essi è costituita da linguaggi di alto livello, che possono essere potenzialmente utilizzati su ogni computer esistente sulla faccia della terra; parecchi di questi sono correntemente implementati sull'Amiga e molti altri lo saranno in un futuro più o meno prossimo. Accanto ai linguaggi evoluti esiste però una serie di linguaggi, ognuno dei quali funziona solo su una ben determinata categoria di computer, che vengono identificati con il nome collettivo di 'linguaggi Assembly'; ogni famiglia di microprocessori possiede il suo personale linguaggio Assembly, in tutto o in parte incompatibile con quelli di tutti gli altri processori. Questa torre di Babele informatica avrebbe decretato la scomparsa di tali linguaggi dall'uso comune, se non fosse stato per un piccolo particolare: l'Assembly, non essendo ne' interpretato nè compilato ma direttamente compreso ed eseguito dal processore, è il linguaggio più efficiente che esista. Si può dimostrare matematicamente che l'Assembly è il più veloce tra tutti i linguaggi passati, presenti e futuri implementabili su un determinato microprocessore, nonché il meno affamato di memoria (fatta eccezione per i linguaggi interpretati ove l'interprete sia residente in ROM). Un altro fattore che ha limitato in concreto il successo dell'Assembly è la diffusa convinzione che sia difficilissimo usarlo: potrete rendervi personalmente conto seguendo questo corso dell'infondatezza di tale pregiudizio. La programmazione in Assembly non richiede di fatto sforzi mentali molto superiori a quelli imposti da linguaggi presunti evoluti come il C o il Forth.

Un po' di nomenclatura

Esistono diversi termini di uso comune nell'informatica la cui conoscenza è consigliabile in generale e d'obbligo per i programmatori in Assembly in particolare: bit, byte, word, long word, RAM e ROM.

BIT: contrazione di BInary digIT che significa cifra binaria; trattasi di un ente capace di assumere due soli possibili valori (o, se preferite, di esistere in uno solo di due possibili stati) chiamati convenzionalmente zero e uno.

BYTE: una sequenza di otto bit. Unità di misura della quantità di informazione, deve la sua popolarità al fatto che i primi personal computers (e molti di quelli attuali) erano a otto bit, cioè la loro memoria era fisicamente suddivisa in gruppi (chiamati parole) di otto bit.

WORD: questo termine possiede due significati distinti: il primo è quello, letterale, di 'parola'; il secondo, caratteristico dell'Assembly MC68000, è quello di 'parola di 16 bit'. LONG WORD: nel caso dell'Assembly MC68000, significa 'parola di 32 bit'.

RAM: Random Access Memory, memoria a accesso casuale (inteso come il contrario dell'accesso sequenziale); trattasi di un dispositivo molto veloce per la memorizzazione di informazioni, presente in ogni computer; qualcuno (assai pochi, in verita') la chiama RWM, Read-Write Memory.

ROM: Read Only Memory, memoria a sola lettura; differisce dalla RAM in un solo aspetto: i dati vengono scritti nella ROM una volta sola (spesso dallo stesso produttore del componente) e non possono essere più nè cancellati nè modificati, ma solo letti. Rispetto alla RAM, la ROM presenta il vantaggio di non perdere i dati immagazzinati quando si spegne il computer.

Organizzazione della memoria

La memoria (sia RAM che ROM) è suddivisa in parole che, nel caso del nostro Amiga, sono di 16 bit; tuttavia, per motivi storici e pratici, può anche essere pensata come suddivisa in byte. I byte sono numerati in modo da poterli distinguere l'uno dall'altro e tali numeri, chiamati 'indirizzi', partono da zero e vanno fino a 16777215 (no, non dovete impararlo a memoria). I primi 524288 byte sono i famosi 512K (1K = 1024 byte: 512x1024 = 524288) di chip RAM che ogni Amiga possiede; i rimanenti indirizzi possono essere occupati dalla ROM, da espansioni di RAM o da dispositivi 'memory

mapped'(ritorneremo in seguito su questo argomento con assai maggiori dettagli). Sulla memoria e' possibile eseguire due operazioni fondamentali, dette lettura e scrittura: nel primo caso noi leggiamo il contenuto di uno o più byte di memoria dopo averne specificato l'indirizzo, nel secondo scriviamo un certo valore in una certa zona della memoria dopo aver specificato sia l'indirizzo sia il nuovo valore, che verrà sostituito al precedente contenuto. Nella quasi totalità dei casi, quando parleremo di lettura o scrittura sarà sottinteso che è il microprocessore, da noi programmato, a effettuare tali azioni. Ogni operazione eseguita sulla memoria viene chiamata 'accesso alla memoria'.

A ogni indirizzo è associato quindi un byte. Le word, al contrario, essendo sequenze di due byte, esistono solo a indirizzi pari: 0, 2, 4, 6 e così via. A esempio, la word 0 è formata dai byte 0 e 1, la word 2 dai byte 2 e 3. Due qualsivoglia word consecutive formano una long word, la quale di conseguenza deve avere un indirizzo pari; a esempio, la long word 0 è formata dalle word 0 e 2 (e quindi dai byte 0, 1, 2 e 3), la long word 2 dalle word 2 e 4 (e quindi dai byte 2, 3, 4 e 5).

Si deduce da quanto detto che le long word sono parzialmente sovrapposte a due a due; non c'è niente di male in ciò, basta ricordarlo sempre. Se tentate di accedere a una word o a una long word specificando un indirizzo dispari il vostro programma si bloccherà provocando una Guru Meditation di tipo 00000003.

Le rappresentazioni binaria ed esadecimale

Chi conosce già questo argomento è ufficialmente autorizzato a saltare questo capitolo. Anni fa le basi diverse da dieci non venivano usate con i linguaggi evoluti, ma già da un po' di tempo chi lavora in C o in Forth trova spesso utile conoscerle; la rappresentazione esadecimale, a esempio, è largamente impiegata nei programmi in C descritti nei manuali dell'Amiga.

È opportuno inoltre considerare il fatto che molti tool per il debugging come monitor e disassembler usano l'esadecimale. Se rifiutate di imparare queste rappresentazioni potrete ugualmente lavorare in Assembly, ma presto o tardi arriverete a un punto in cui rimpiangerete di non averle apprese.

Tutti conoscono la numerazione in base dieci: ogni numero è composto da una stringa di cifre, ognuna delle quali può assumere dieci possibili valori (da zero a dieci meno uno); la cifra più a destra (se

per semplicità ci limitiamo agli interi) è quella meno significativa, comunemente detta cifra delle unità; ogni altra cifra deve essere moltiplicata per il valore del posto in cui si trova, tenendo presente che il valore della posizione aumenta di un fattore dieci ogni volta che ci si sposta di un posto e sinistra.

Quando si passa a un altro sistema di numerazione, a esempio in base due, basta sostituire la parola due alla parola dieci nelle frasi precedenti. A esempio, 4321 significa 1+2x10+3x10x10+4x10x10x10, mentre $4321 (in futuro useremo il segno '$' per introdurre un numero in esadecimale e il segno '%' per il binario) significa 1+2x16+3x16x16+4x16x16x16 e infine %4321 non ha senso perché in binario le cifre da due in su non esistono, proprio come in decimale non esistono le cifre da dieci in su: la cifra di massimo valore e' il nove. Un esempio di numero binario è %1101 che significa 1+0 x 2+1 x 2 x 2+1 x 2 x 2 x 2=13.

Naturalmente, se in binario esistono solo due cifre e in decimale ne esistono dieci, è facile intuire che in esadecimale ne esistono sedici. Le cifre da dieci a quindici vengono rappresentate con le lettere dell'alfabeto da A a F (A=10, B=11, C=12, D=13, E=14, F=15): a esempio $2A3C=12+3 x 16+10 x 16 x 16+2 x 16 x 16 x 16=10812.

L'utilità del sistema esadecimale è dovuta al seguente fatto: i computer lavorano in binario, ma tale notazione è eccessivamente scomoda per un essere umano a causa della sua prolissità (a esempio 200 diventa %11001000 in binario); la notazione esadecimale è invece molto compatta ed è davvero molto facile convertire un numero dal binario all'esadecimale e viceversa.

Il metodo consiste nel suddividere il numero binario in gruppetti di quattro cifre e convertirli separatamente in singole cifre esadecimali: %1100101100001 diventa 1 1001 0110 0001 e quindi $1961. Ometto la dimostrazione, peraltro elementare, di questa proprietà che deriva essenzialmente dall'essere il sedici una potenza intera del due. Il passaggio dal binario o dall'esadecimale al decimale non è invece altrettanto immediato. In generale, non avvertirete quasi mai la necessità di convertire numeri manualmente: ci penserà l'assembler a farlo per voi.

Cos'è un microprocessore

Il microprocessore è il cuore di ogni personal computer, o meglio ne è il cervello. Esso può essere considerato alla stregua di un computer in miniatura, capace di eseguire istruzioni molto semplici e dotato di qualche decina di byte di RAM interna ad accesso molto veloce, suddivisa in unità chiamate registri.

Questi ultimi sono concettualmente molto simili alle memorie di una calcolatrice in quanto sono presenti in numero limitato e vengono individuati tramite nomi convenzionali stabiliti una volta per sempre, al contrario delle variabili di un linguaggio evoluto il cui numero e i cui nomi possono variare da programma a programma; tuttavia, mentre le memorie di una calcolatrice sono appositamente progettate per immagazzinare numeri in virgola mobile, i registri sono dispositivi capaci di memorizzare ognuno n bit di informazione il cui esatto significato può variare in maniera assai ampia: nella maggior parte dei casi questi bit verranno usati per rappresentare un numero intero, ma nessuno impedisce di costruire programmi che li interpretano invece come un numero in virgola mobile o addirittura di utilizzarli indipendentemente l'uno dall'altro come una serie di flag.

Questa affermazione potrà stupire tutti quei programmatori che hanno sempre fatto uso di quei linguaggi evoluti in cui esistono tipi di dati rigidamente distinti e incompatibili tra di loro: se una variabile è stata dichiarata a esempio come intera non potrà essere mai utilizzata per memorizzare un numero in floating point e viceversa.

Questa regola, rispettata in quasi tutti i linguaggi evoluti esistenti, viene invece clamorosamente violata dall'Assembly in cui i tipi di dati sono virtualmente inesistenti, o, più esattamente, esistono solo nella mente del programmatore e nella struttura generale del programma. Un esempio chiarirà meglio questo concetto.

Supponiamo di aver posto delle informazioni nei registri D1 e D2 e di far eseguire poi al processore l'istruzione ADD D1,D2; il 68000 preleverà i contenuti dei due registri, li interpreterà come numeri interi a 16 bit (cioè, come vedremo più avanti, di valore compreso tra -32768 e +32767: ciò che in C viene chiamato short integer) e li sommerà, ponendo il risultato, anch'esso intero, nel registro D2. Tutto ciò accadrà indipendentemente dalla natura dei dati che abbiamo inserito nei registri D1 e D2, in quanto l'istruzione ADD D1,D2 forza il processore a partire dal presupposto che i summenzionati registri contengono interi a 16 bit.

Se invece l'istruzione fosse stata ADD.B D1,D2 il processore avrebbe sommato i contenuti dei registri considerandoli interi a otto bit (il suffisso .B significa Byte; in C verrebbe chiamato char), mentre se il 68000 avesse eseguito un BSET D1,D2 avrebbe settato, ossia posto a uno, il D1-esimo bit del registro D2, interpretando quindi il contenuto di D2 come una serie di bit indipendenti.

Forse penserete che tutto questo provochi una gran confusione, ma non è così: al programmatore è semplicemente consentita la massima libertà nel manipolare i dati e nel commettere errori, e proprio questa libertà rappresenta uno dei maggiori punti di forza dell'Assembly. Supponiamo di avere un intero nel registro D1 e di voler porre in D2 il valore 2^D1; il metodo più semplice per raggiungere tale risultato consiste nell'azzerare tutti i bit di D2 tranne il D1-esimo, che deve essere invece settato. A esempio 2^3=8=%00001000; dal momento che il bit meno significativo, ovvero quello più a destra, è per convenzione il bit numero zero e dato che la numerazione dei bit procede da destra a sinistra si vede subito che l'unico bit settato è proprio il numero tre.

Saranno quindi sufficienti due istruzioni: CLR D2, che azzera D2 considerandolo uno short integer, e BSET D1,D2, che, come spiegato in precedenza, setta il D1-esimo bit di D2, considerando quindi quest'ultimo come una stringa di bit indipendenti. Ecco quindi che uno coppia di istruzioni apparentemente contraddittorie, attribuendo ognuna a D2 un diverso significato, raggiunge invece lo scopo prefisso.

Un tipico programma in Assembly, per quanto possa sembrare strano, letteralmente pullula di trucchi di questo genere.

Ogni processore possiede un registro chiamato PC (Program Counter, contatore di programma; qualcuno lo chiama IP, Instruction Pointer, puntatore alle istruzioni) che contiene l'indirizzo dell'istruzione in fase di esecuzione. Il processore infatti non fa altro che ripetere le seguenti operazioni in un ciclo senza fine: legge dalla memoria il codice di una singola istruzione, incrementa il PC in modo che punti alla prossima, decodifica l'istruzione e la esegue. Il PC serve quindi a ricordare al processore a che punto del programma è arrivato; una qualunque istruzione di salto, condizionato o non, agisce semplicemente cambiando il contenuto del PC.

Un altro registro sempre presente è lo SP (Stack Pointer, puntatore allo stack); lo stack è un'area di memoria destinata a immagazzinare gli indirizzi di ritorno delle subroutine, anche se spesso viene utilizzato anche per altri scopi. Ogni chiamata di subroutine decrementa di quattro unità lo SP, trasferisce il contenuto attuale del PC nella long word puntata dallo SP e in-

fine pone nel PC l'indirizzo della subroutine da eseguire. Ogni istruzione di ritorno ritrasferisce nel PC il valore puntato dallo SP e incrementa quest'ultimo di quattro unità, ripristinando così le condizioni esistenti prima della chiamata di subroutine. Questo argomento, data la sua grande importanza, verrà trattato in futuro più estesamente.

Schema dei registri del processore.

Il modello di programmazione del 68000

In fig.1 è illustrato lo schema dei registri del 68000. Ogni rettangolo rappresenta un registro, il cui numero di bit è graficamente simboleggiato dalla lunghezza del rettangolo. Potete quindi vedere che il 68000 possiede otto registri dati (data registers) i cui nomi sono formati dalla lettera D seguita da un numero da zero a sette, più otto registri indirizzi (address registers) i cui nomi differiscono da quelli dei registri dati unicamente per la presenza della lettera A al posto della D.

In teoria i registri indirizzi dovrebbero essere usati per immagazzinarvi, come si può facilmente immaginare, degli indirizzi, mentre i registri dati dovrebbero servire genericamente a contenere qualunque tipo di dato che non sia un indirizzo.

Dal punto di vista pratico, l'unica differenza tra questi due tipi di registri consiste nel fatto che molte istruzioni, come a esempio le precedentemente introdotte CLR e BSET, agiscono solo sui registri dati, mentre qualche altra istruzione, come a esempio LEA (Load Effective Address), agisce solo sui registri indirizzi; oltre a ciò, solo questi ultimi possono essere usati come puntatori. Il registro A7 è un po' sui generis in quanto coincidente con lo stack pointer, il tutto per motivi di praticità: in questo modo infatti lo SP può essere gestito con la stessa flessibilità di un registro indirizzi. Un'altra caratteristica che rende particolare tale registro è la presenza di un suo doppione chiamato A7', che funge da stack pointer secondario; il 68000 possiede in effetti due stack pointer e il motivo di questa stranezza apparirà chiaro dopo aver introdotto il concetto di modo operativo.

Il processore, in ogni dato istante, può trovarsi in uno di due possibili stati o modi operativi chiamati *modo supervisore* altrimenti noto come *privilegiato* e *modo utente*. Quando il processore funziona in modo supervisore possiede il controllo completo del computer, mentre se si trova in modo utente non può eseguire determinate istruzioni che vengono pertanto definite *privilegiate*. Tutto ciò serve, nei sistemi multitasking, a impedire che un task impazzito a causa di un bug possa paralizzare il computer eseguendo un'istruzione privilegiata (che, se fosse eseguita da un programma malfunzionante, metterebbe a soqquadro il sistema operativo).

Bisogna onestamente riconoscere che si tratta di una misura di sicurezza del tutto inadeguata: ogni possessore di Amiga sa che se uno dei programmi che girano in quel momento sul computer salta a causa di un bug non vi è alcuna speranza di salvare gli altri task: la ben nota Guru Meditation tronca qualunque tentativo di illudersi in tal senso. Sappiate comunque che esistono due diversi stack, uno dei quali viene usato in modo supervisore, l'altro in modo utente; quasi tutti i programmi che scriverete agiranno completamente in modo utente.

Il PC lo conoscete già; lo SR (Status Register, registro di stato) è l'unico registro a 16 bit del processore e contiene alcune informazioni circa il corrente modo operativo, la priorità dell'interrupt, l'attivazione del funzionamento in modo trace e una serie di cinque flag che vengono influenzati dal risultato dei calcoli che il processore sta svolgendo. Tutti i registri del 68000, fatta eccezione per lo SR, sono a 32 bit e proprio questo fatto ha causato la nascita di quella leggenda in base alla quale il 68000 sarebbe un microprocessore a 32 bit. Tale leggenda risulta infondata, non essendo la ALU (Arithmetic Logic Unit) nè il bus del processore a 32 bit, ma non vi è comunque dubbio che la dimensione e soprattutto il numero dei registri del 68000 siano alcuni tra i motivi principali che lo rendono uno dei più efficienti processori a 16 bit che siano mai esistiti, immensamente più potente a esempio dell'8088 tipico del mondo MS-DOS.

CORSO

Prima puntata di un corso sull'AmigaBASIC: l'interprete BASIC dell'Amiga

di Paolo Russo

Che il BASIC sia il linguaggio di programmazione più diffuso sui personal computer non è un mistero per nessuno. Purtroppo il BASIC detiene anche un altro e meno desiderabile primato, quello del linguaggio più denigrato del mondo; nessun altro linguaggio ha mai suscitato un più vasto mare di polemiche, nè è stato mai fatto oggetto di strali da parte di tanti e tanto autorevoli personaggi del mondo dell'informatica. Sembra infatti molto diffusa l'abitudine di pensare all'informatica come a una medaglia con due facce che potremmo definire del bene e del male, simboleggiate rispettivamente la prima dal Pascal e dai suoi derivati e la seconda dal BASIC, acronimo che, ricordiamolo ancora una volta, significa proprio Beginners All-Purpose Symbolic Instruction Code, ossia linguaggio polivalente per principianti. Bisogna onestamente ammettere che la prima versione del BASIC era davvero atroce, ma da allora molta acqua è passata sotto i ponti.

Il fatto che il BASIC fosse molto diffuso sui personal lo ha reso un linguaggio vivo come pochi altri, ovvero un insieme di dialetti in indipendente, continua e rapida evoluzione. Il BASIC è quindi diventato sempre più serio e strutturato, al punto da rendere anacronistiche le critiche che ancor oggi gli vengono mosse, senza perdere la sua grande dote: la enorme semplicità d'uso, che nessun altro linguaggio è mai riuscito a eguagliare. L'Amiga è un computer innovativo come pochi altri, e il suo dialetto BASIC deve essere, e in certa misura è, almeno altrettanto innovativo.

Questa serie di articoli illustrerà tutte le caratteristiche peculiari dell'AmigaBASIC: la strutturazione, il controllo del mouse, dei menu a discesa, della grafica, della sintesi sonora e vocale e chi più ne ha più ne metta.

I tipi di dati

La funzione principale di un computer consiste nella manipolazione di informazioni di ogni genere, che devono essere presenti nella memoria della macchina in una forma ben precisa. Decenni di esperienza programmatoria mondiale suggeriscono che le due categorie di dati più importanti sono i numeri e le stringhe alfanumeriche: con questi due tipi di dati fondamentali si può rappresentare qualunque informazione in maniera abbastanza semplice. A esempio, 123 è un numero mentre "salve" è una stringa.

Il fatto che numeri e stringhe siano tipi di dati differenti significa che la variabili che li memorizzeranno dovranno avere una struttura interna diversa e che le operazioni che sarà lecito effettuare su dati del primo tipo non saranno applicabili a quelli del secondo e viceversa. Potete a esempio moltiplicare due numeri, ma non due stringhe; queste ultime potranno invece, a differenza dei primi, essere concatenate o letteralmente ridotte a fettine in più di un modo.

Ogni interprete BASIC esistente supporta perlomeno i due tipi di dati appena menzionati, ma la maggior parte dei dialetti in circolazione consente la distinzione tra numeri interi e reali (o, più propriamente, in virgola mobile). Gli interi non sono, a rigor di termini, assolutamente indispensabili, potendo essere sostituiti dai reali in virtualmente ogni applicazione, ma forniscono al sistema una maggiore versatilità ed efficienza: miriadi di applicazioni sono basate interamente o quasi sull'aritmetica intera ed è risaputo che gli interi consumano meno memoria e possono essere manipolati più velocemente: anche qualora aveste a disposizione un coprocessore aritmetico, le operazioni eseguite sugli interi sarebbero sempre più veloci delle altre. In taluni casi l'uso degli interi consente perfino di ridurre gli errori di calcolo.

Quanto detto sopra sui tipi è valido per il BASIC in generale; come si comporta a tale riguardo il nostro AmigaBASIC ?

L'interprete BASIC dell'Amiga possiede, per la gioia dei programmatori di tutto il mondo, ben cinque tipi di dati diversi. Uno di essi è rappresentato dalle stringhe, l'uso delle quali all'interno dell'AmigaBASIC non diverge molto dalla norma. I rimanenti quattro tipi sono tutti numerici: reali in singola precisione, reali in doppia precisione, interi corti e interi lunghi. I due generi di numeri reali differiscono per il numero di byte utilizzati per la loro rappresentazione interna: quattro byte per la singola precisione e otto per la doppia.

La prima conseguenza di questa apprezzabile differenza, con particolare riferimento alla risoluzione doppia, consiste nella drastica riduzione degli errori di calcolo, ovviamente motivata dall'aumento

DI
AMIGA
BASIC

delle cifre significative nei dati iniziali e in ogni risultato intermedio; il secondo e meno piacevole effetto è dato dal raddoppio della quantità di RAM necessaria per i dati, fatto questo particolarmente preoccupante in presenza di vasti array; la terza e infine ultima conseguenza consiste nella diminuzione della velocità di elaborazione.

La scelta tra questi due tipi di dati può quindi essere influenzata da diversi fattori ed è bene che ogni programmatore ne tenga conto. Se invece il programmatore rifiuta di pensare a simili trascurabili dettagli, per pigrizia o semplice fretta, penserà l'interprete a prendere coraggiosamente posizione in sua vece: dal momento che la velocità e la compattezza sono spesso ritenute dei requisiti di maggiore importanza della precisione, il nostro BASIC ci fornisce sempre le variabili in singola precisione ovunque non sia specificato altrimenti.

Anche gli interi, come sopra accennato, esistono in due varietà: corti o lunghi. I primi devono avere un valore compreso tra -32768 e +32767; qualunque tentativo di assegnare loro un valore al di fuori di tali limiti provocherà il blocco del programma oltre a una alquanto seccante segnalazione di errore di overflow. Gli interi lunghi risentono di una limitazione ben più sopportabile, consentendo di gestire numeri il cui valore assoluto può tranquillamente raggiungere un paio di miliardi. Questa maggiore flessibilità ha un prezzo, che è in fondo molto simile a quello che bisogna pagare passando dai reali in singola precisione a quelli in doppia: maggiore consumo di memoria (quattro byte anziché due) e abbassamento della velocità di calcolo.

L'AmigaBASIC decide il tipo di una variabile basandosi sull'ultimo carattere del nome della stessa, come risulta dalla tabella 1; in assenza di particolari suffissi la variabile è considerata reale in singola precisione per default. Supponiamo però che tutte o quasi le variabili di un programma siano, a esempio, intere; sarebbe alquanto seccante dover appendere il carattere '%' alla fine di ogni nome. Esiste per fortuna una simpatica scorciatoia: e' sufficiente inserire un DEFINT A-Z in testa al programma per far sì che l'interprete consideri intere tutte le variabili il cui nome è privo di particolari suffissi.

Se invece le variabili dovessero essere tutte reali in doppia precisione potremmo usare un DEFDBL A-Z, come illustrato nella tabella 1. Un altro esempio: se desideriamo che tutte le variabili siano reali in doppia precisione, a eccezione di quelle il cui nome inizia per I, J, K, L, M o N, che dovranno invece essere considerate intere (queste lettere in matematica vengono sovente adoperate per indicare degli indici, i quali sono solitamente interi), basterà un DEFDBL A-Z seguito da un DEFINT I-N: il primo comando definirà tutte le variabili (nessuna esclusa) come reali, mentre il secondo ridefinirà come intere quelle che noi vogliamo siano tali.

Ogni DEFxxx può annullare o modificare le convenzioni imposte da ogni precedente DEFxxx. La cosa curiosa, nonché interessante, è che lo statement DEFxxx deve essere eseguito per avere effetto, al contrario di altri comandi, come a esempio DATA, ai quali si richiede semplicemente di essere presenti da qualche parte nel listato, anche se si trovano in una sezione che non verrà mai eseguita. È perfino possibile immaginare di eseguire un DEFxxx condizionato in fase di runtime, il tutto allo scopo di garantire la massima flessibilità. Un esempio di una possibile applicazione? Potreste realizzare un programma che svolge calcoli intensivi su vasti array di reali, circostanza che si verifica sempre quando si ha a che fare con il calcolo matriciale, e potreste far sì che sia il programma stesso a chiedere all'utente qual'è la precisione che deve essere impiegata nei calcoli, e se l'utente esprime il desiderio di adoperare il massimo numero di cifre significative il programma eseguirà un DEFDBL A-Z, mentre salterà tale statement in caso contrario.

A ogni modo, un programma realizzato con questa tecnica non potrebbe mai essere compilato; in genere i compilatori si aspettano che queste definizioni globali di tipo vengano utilizzate una volta sola, all'inizio del programma, e non vengano più ritoccate.

Normalmente l'AmigaBASIC converte automaticamente i numeri da un tipo all'altro quando si assegnano quantità di un certo tipo a variabili di tipo diverso: eseguendo un A%=3.2 in A% finirà il valore intero tre. Una vistosa (e dolorosa) eccezione a questa regola consiste nel passaggio di parametri ai subprogram, come vedremo in seguito. È comunque doveroso attirare l'attenzione del lettore su di un antipatico problema: supponiamo che il computer esegua l'istruzione A# =1/3; cosa finirà in A# ? Si accettano scommesse. Il simbolo '#' impone al BASIC di considerare A# come una variabile reale in doppia precisione e chiunque sarebbe quindi tentato di pensare che A# contenga adesso il valore 'un terzo' in doppia precisione: ebbene, non è così. I numeri 1 e 3 sono interi; il computer, dovendo eseguire su di essi una divisione, che è notoriamente un'operazione da eseguirsi sui reali, li converte automaticamente, bontà sua, in reali (in singola precisione) prima di applicarvi la suddetta operazione. Il risultato, ottenuto operando su reali in singola precisione, sarà anch'esso del medesimo tipo. Infine tale risultato viene convertito nel formato della doppia precisione (ma questo, di per sè, non ne aumenta il numero di cifre significative) e assegnato ad A#, che conterrà quindi il valore 1/3 in singola precisione. Per ottenere realmente un risultato in doppia precisione avremmo dovuto eseguire A# =1#/3#, costringendo quindi il pigrissimo AmigaBASIC a svolgere effettivamente la divisione in doppia precisione. Provare per credere.

TIPO	ESEMPIO	DEFxxx	X-> STR	STR-> X	NUM-> X
string	var$	DEFSTR			
short	var%	DEFINT	MKI$	CVI	CINT
long	var&	DEFLNG	MKL$	CVL	CLNG
float	var!	DEFSNG	MKS$	CVS	CSNG
double	var#	DEFDBL	MKD$	CVD	CDBL

La strutturazione

Le due istruzioni maggiormente odiate e disprezzate dai fanatici della programmazione strutturata sono le ben note GO-

TO e GOSUB. La principale causa di cotanto risentimento deriva da tre motivi: prima di tutto il GOTO consente di uscire dalle strutture e zigzagare a volontà in lungo e in largo nel programma, cosa questa che lascia disorientati molti programmatori; in secondo luogo non è facile capire quale sia la funzione dei vari GOTO e GOSUB perché i numeri di linea da cui queste istruzioni sono solitamente seguite non sono dotati di particolari facoltà autoesplicative; in terzo luogo i GOSUB non consentono di passare parametri agevolmente alle subroutine.

Per quanto riguarda il primo motivo, il problema qui deriva da una libertà che ai fan di Niklaus Wirth (il padre del Pascal) appare eccessiva, ma è opportuno ricordare che, per quanto ogni persona al mondo sia abituata a strutturare i propri pensieri, non esiste alcuna legge che obblighi a farlo allo stesso modo del signor Wirth; d'altra parte, non c'è nemmeno una legge che renda obbligatorio l'impiego dei GOTO al posto delle strutture FOR - NEXT, WHILE - WEND e IF - THEN - ELSE - ELSEIF - END IF, che sono ovviamente supportate dall'AmigaBASIC. Ognuno è, in sostanza, libero di fare in AmigaBASIC ciò che preferisce e di farlo nel modo che più gli aggrada.

Per quanto riguarda la scarsa eloquenza dei numeri di linea, tale problema viene brillantemente risolto con l'introduzione delle label alfanumeriche in sostituzione dei non più obbligatori numeri di linea: in altri termini potete scrivere GOSUB TracciaGrafico in luogo di GOSUB 2537.

Rimane la questione del passaggio dei parametri alle subroutine, che viene parzialmente risolto con la creazione di un nuovo ente: il subprogram. Tale struttura vorrebbe rappresentare l'equivalente BASIC delle procedure, tipiche dei linguaggi Pascal-like. Un subprogram viene introdotto dalla dichiarazione SUB nome (fpar1, fpar2... fparn) STATIC, seguita dal corpo del sottoprogramma e infine dalla parola chiave END SUB, mentre per richiamarlo si possono utilizzare due diverse sintassi: CALL nome (par1, par2... parn) oppure semplicemente nome par1, par2... parn, sottintendendo quindi la parola chiave CALL nonché le parentesi di delimitazione dei parametri.

Par1 significa parametro numero uno, mentre fpar1 sta per parametro formale numero uno; la differenza tra i due consiste nel fatto che, mentre par1 è una qualunque espressione che compare nella chiamata del subprogram, fpar1 è una variabile che esiste solo all'interno del subprogram e alla quale, al momento della chiamata, viene assegnato il valore di par1. Tutte le variabili che compaiono all'interno di un determinato subprogram sono isolate da quelle del programma principale nonché de quelle di ogni altro subprogram, nel senso che due variabili aventi lo stesso nome, ognuna delle quali appartiene a un diverso subprogram, possono tranquillamente avere due valori diversi. Nella maggior parte dei casi ciò è da reputarsi un fatto positivo, in quanto questa precisa demarcazione previene l'insorgere di uno dei più universalmente diffusi tipi di bug, ossia l'attribuzione dello stesso nome a due variabili concettualmente distinte.

A esempio un subprogram può contenere una linea come FOR I=1 TO 10, mentre il programma principale può richiamare tale subprogram cinque volte di seguito con l'ausilio di un altro ciclo FOR - NEXT basato anch'esso sull'uso di una variabile indice di nome I, senza che questo provochi la minima collisione. Possono tuttavia esistere dei casi, che nella pratica sono in realtà assai frequenti, nei quali è opportuno che il subprogram condivida alcune variabili (tipicamente array) con il programma principale; si può allora ricorrere alla direttiva SHARED var1,var2...varn che, inserita subito dopo la definizione di un subprogram, gli consente l'accesso alle variabili globali elencate; non per nulla 'shared'in inglese significa 'condiviso'.

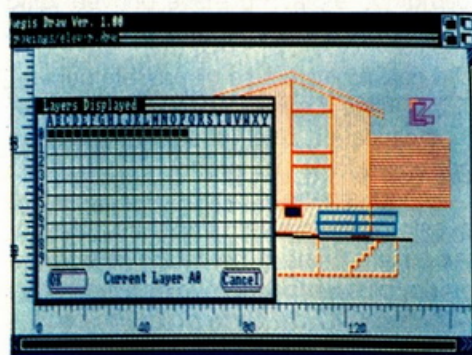
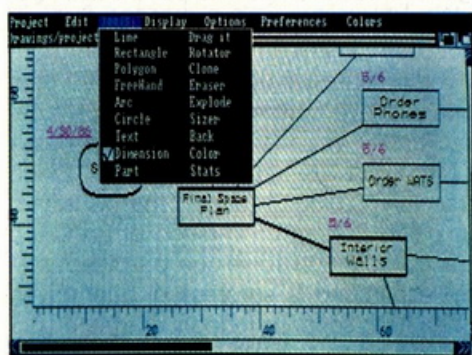
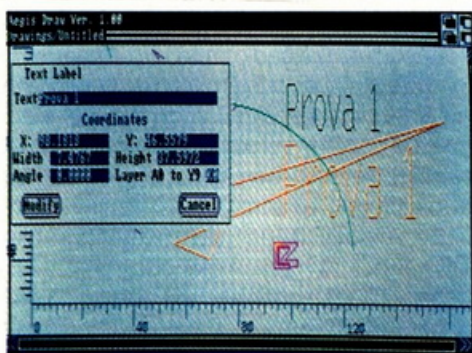
Se poi l'array A() deve essere comune a tutti i subprogram il metodo più pratico consiste nel dimensionarlo in maniera alquanto particolare, tramite un DIM SHARED A(...). Sicuramente molti appassionati di strutturazione non gradiranno molto l'esistenza della direttiva SHARED, considerata poco elegante per non dire peggio, ma in fondo è tutta una questione di gusti personali e di stili di programmazione: nessuno è obbligato a farne uso, ma miriadi di programmatori dotati di senso pratico in seguito alle prime esperienze con l'AmigaBASIC ne apprezzeranno di certo immensamente la presenza.

Qualcuno potrà a questo punto pensare che l'AmigaBASIC sia il linguaggio perfetto, generosamente donato all'umanità da chissà quale divinità dell'Olimpo come ricompensa di innumerevoli sacrifici di innocenti capretti; beh, la situazione non è proprio esattamente questa, l'AmigaBASIC proviene più prosaicamente dai programmatori della Microsoft, che di errori ne commettono anche loro a bizzeffe, come qualunque altro essere umano. È giunto il momento di discutere a proposito dei limiti e dei difetti dei subprogram.

Innanzitutto il meccanismo di passaggio dei parametri è alquanto deprimente: si tratta di un procedimento estremamente lento (evitate di richiamare i subprogram nei tratti di programma ai quali si richiede una particolare velocità) che non effettua alcun tentativo di conversione dei parametri da un tipo all'altro.

Se cioè definite un SUB traccia(x,y) STATIC e poi lo richiamate con un CALL traccia(100,50) otterrete una splendida segnalazione di errore come 'ovvia' conseguenza dell'aver definito per default i parametri formali x e y come reali e dell'aver successivamente tentato di assegnare loro i valori rispettivamente di 100 e 50 che, fino a prova contraria, sono da considerarsi interi. Vi viene gentilmente concessa l'alternativa tra l'utilizzo di un CALL traccia(100.0,50.0) (augh!, ma qui siamo in pieno Fortran!), l'impiego di un CALL traccia(100!,50!) (forse quei punti esclamativi esprimono la sorpresa di quei poveri interi nello scoprirsi indegni di essere considerati reali), la sostituzione dell'originale definizione del subprogram con SUB traccia(x%,y%) STATIC (ma attenzione che così protesterà se gli passate dei reali...) e infine il suicidio con una mitragliatrice caricata a pixel (di bassa risoluzione per aprire ferite più larghe).

Scherzi a parte, è piuttosto evidente che questo difetto davvero imperdonabile dovrà essere prima o poi risolto in qualche modo; più che in una nuova release dell'AmigaBASIC da parte della Microsoft è probabile che sia più sensato auspicare la creazione di un compilatore che rimedi a questa clamorosa svista eseguendo una conversione implicita ovunque necessario, come del resto accade sempre nelle assegnazioni alle variabili e nei calcoli intermedi: ciò non sarebbe davvero difficile a realizzarsi.

Altri problemi inerenti ai subprogram, oltre alla precedentemente citata lentezza, sono dati a esempio dal fatto che i DEFxxx agiscono contemporaneamente sul programma principale come su ogni subprogram esistente, dall'impossibilità di richiamare tramite un semplice GOSUB le subroutine definite nel programma principale (rendendo quindi impossibile la loro condivisione tra i vari subprogram), la loro minore elasticità rispetto alle funzioni multistatement, dolorosamente assenti e indegnamente sostituite dalle più classiche DEF FN e l'impossibilità di dimensionare array o inizializzare variabili locali se non facendo ricorso a trucchi ignobili di cui, non temete, vedrete comunque degli esempi in futuro.

AEGIS DRAW:

Breve escursione all'interno di un programma per il disegno con vocazione decisamente professionale

di Fabio Biancotto

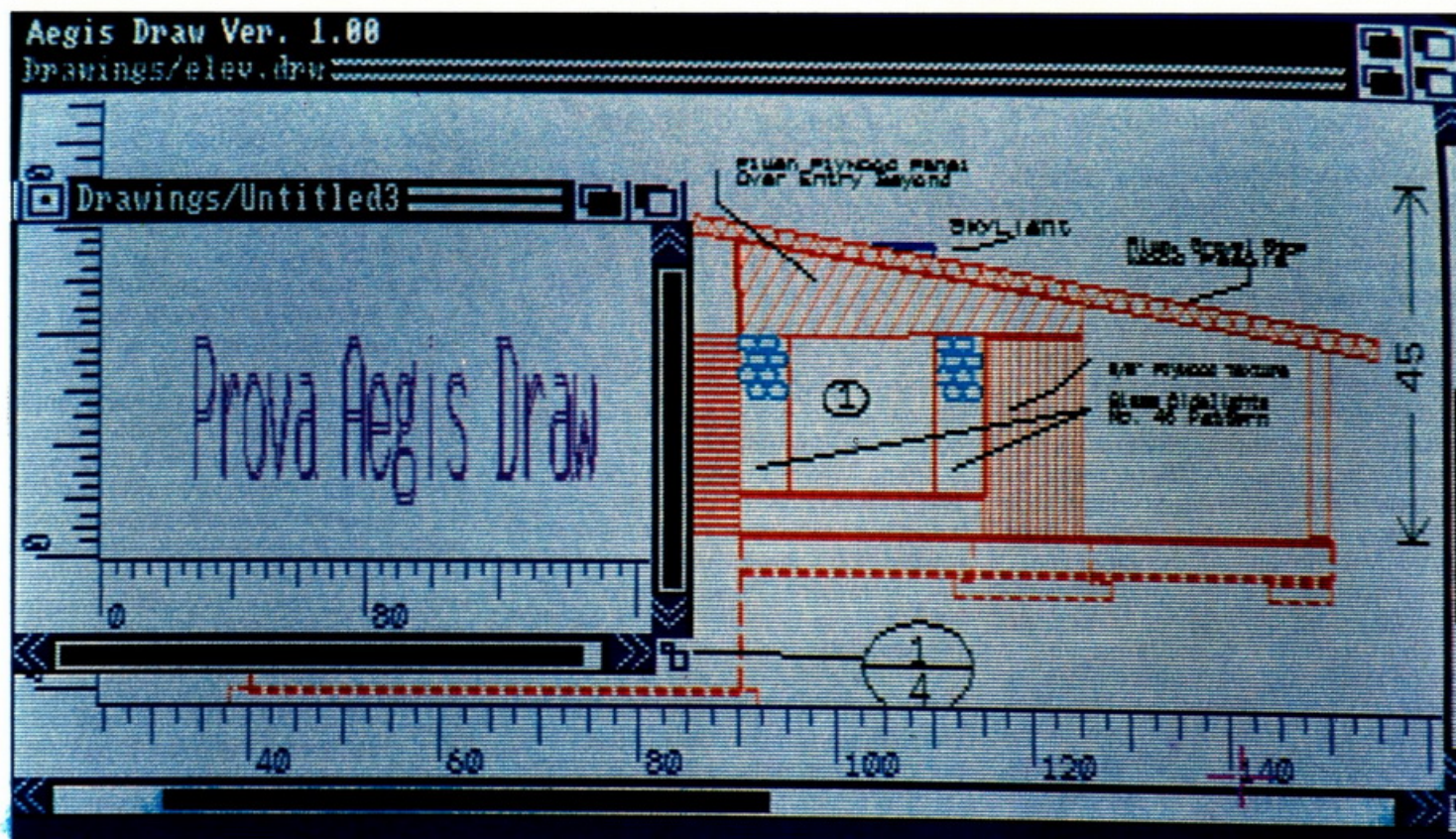
Più o meno tutti hanno un programma grafico. Che sia l'Electronic Arts'Deluxe Paint o Aegis Images, della Aegis Development, o qualche altro package del quale non siamo al corrente, questi package sono sofisticati e forniscono, a colui che li usa una volta ogni tanto o al più esperto, una enorme gamma di possibilità.

Bit-map e object-oriented

Che cosa dà un CAD per giustificare il suo costo? Prima di tutto, chi lo usa deve capire la differenza tra i grafici a bit-map e quelli di tipo object-oriented. I package più semplici, prima citati, si affidano ad una gestione delle immagini di tipo bit-map; pertanto, il programma stesso, non ha, o se ce l'ha è sempre in minima parte, sotto controllo le immagini da lui generate: quali cerchi, quadrati o linee.

Dal momento in cui qualcosa è sullo schermo, viene trattato come una massa di puntini singoli. Pertanto i puntini che formano il cerchio sono esattamente gli stessi che formano il quadrato che vi si sovrappone (per ciò che riguarda il Deluxe Paint), quindi il programma non può avere un'immagine precisa, separata delle due immagini.

Aegis Draw, programma sviluppato dall'Aegis, è invece un programma del tipo object-oriented. Ciò significa che esso tratta ogni figura che disegni come una entità a sè stante, individuale. Un CAD, quindi, memorizza un oggetto grafico, come per esempio un cerchio, come composto da punto centrale di riferimento e un raggio; dopodiché usa un processo matematico per disegnarlo in modo che lo si possa riconoscere come un cerchio.

Con Aegis Draw si può "acchiappare" un cerchio o un quadrato e spostarlo in

ARRIVANO I CAD PER AMIGA

un qualsiasi punto dello schermo, senza che a causa di tale operazione si interferisca con l'altrea immagine; anche se questi sono sovrapposti.

Un'altra importantissima differenza che mette ancora di più in evidenza le diverse vocazioni di questi due sistemi, è che , a causa del particolare sistema di riconoscimento delle immagini adottato dai CAD (di tipo matematico), gli oggetti da questi ultimi generati, possono essere mossi con una precisione elevatissima e totalmente arbitraria.

Essenzialmente, ciò significa che quando si invia il proprio lavoro verso un dispositivo dotato di migliore risoluzione grafica di quella dello schermo, per esempio un plotter, il risultato sarà perfetto tanto quanto il dispositivo permettera. I cerchi saranno il più "circolari" possibile, le linee diagonali saranno il meno frastagliate possibile. Pertanto, di solito, con questo sistema si riuscira ad ottenere sempre la migliore risoluzione possibile.

Inoltre, a causa del fatto che un package di tipo object-oriented tratta gli elementi grafici come espressioni matematiche, ci si potrà "infilare" in un punto particolare di un'immagine e lì aggiungervi solamente dei dettagli.

Ciò significa che, a ragione, si potrà pensare di avere qualcosa come un quadro del sistema solare, e di essere capaci di zoommare dentro una testata di un giornale, retto da un'immagine, grafica appositamente scalata (in scala), di una persona in piedi sulla superficie del pianeta terra. Bello vero.

Come è facilmente intuibile, l'alto costo è quindi determinato da tutte queste possibilità. Comunque, generalmente, i package grafici object-oriented sono più lenti di quelli a bit-map: infatti, ogni qualvolta si disegni qualcosa sullo schermo, per visua-

lizzare tale immagine, possono essere necessari migliaia di calcoli .

Sistema di valutazione

Nel nostro personale sistema di valutazione, vengono considerati 3 fattori base, e sono valutati in una scala da uno a dieci; inoltre c'è più il nostro "punteggio di impressione" (da una a cinque stelle) che esprime la nostra personalissima opinione totalmente svincolata da analisi fattuali. I tre sistemi di base sono:

SAF: Fattore dello Stato dell'Arte, che indica essenzialmente di che grado il programma si avvantaggi delle possibilità offerte dall'Amiga allorché lo si compari con altri programi simili. Indica anche il livello di innovazione portato dal programma. Questo però non è un segno sufficiente affinché il programma venga considerato buono per forza. È possibile per un package essere SAF senza avere molta sostanza. Al nostro daremo comunque un voto pari a 7.

Velocità: abbastanza ovvia l'interpretazione, nonstante questa sia più una sensazione generale che un segno di bontà del prodotto confermata da una prova tramite un bennchmark. Gli daremo un voto pari a 5.

Prestazioni: il programma è stato fatto molto bene, anche rispetto le aspettative . Per esempio un package grafico con nessuna possibilità di stampare non sarebbe "versatile" a meno che, non fosse stato disegnato per l'output o non ne avesse necessità alcuna. Gli daremo 7.

Punteggio di impressione: la nostra opinione, prodotta, come già detto, non tanto da prove oggettive ma più che altro da sensazioni d'uso, dà una valutazione complessiva del package pari a tre stelle.

Aegis Draw è un CAD di basso livello versatile e potente, ma con alcune leggere imperfezoni, anche se porta in sè almeno 2 o 3 dettagli innovativi. È una applicazione del tipo WorkBench: e ciò significa che altri programmi possono funzionare nello stesso momento; con le dovute limitazioni imposte dalla memoria.

Raffrontato ai tipi di Draw package attualmente sul mercato, dal Mac Draft per Macintosh, di Innovative Data Design, Inc., e In A Vision per l'IBM, di Micrographix, Pc, l'Aegis Draw si stacca da loro mirabilmente. Un confronto punto per punto è fuori questione in questa sede, ma elencheremo invece i tool disponibili nell'Aegis Draw e alcune delle scelte che possono essere fatte.

Faremo inoltre solo alcuni raffronti superficiali fra Aegis Draw altri package equivalenti.

I tool per disegnare

Prima di tutto, alcuni cenni sulla bontà dell'impostazione: i programmatori dell'Aegis hanno fatto un buon lavoro nel permettere a chi usa il computer di poter cambiare idea in qualsiasi momento del proprio lavoro. Per selezionare un tool, si usa il tasto che richiama i menu, e da lì si effettuano le proprie scelte.

Una volta che si inizia a fare qualcosa, come disegnare una linea o un riquadro, si prema il tasto di selezione per indicare un punto di partenza, come per esempio il centro di un cerchio dopodiché si muoa il mouse fino a che si avrà ottenuto la forma desiderata (per esempio il cerchio raggiunge il corretto raggio), poi una seconda pressione del tasto completerà l'operazione.

Premere il pulsante per il menu durante un qualsiasi momento dell'operazione sopra indicata, significa la totale cancellazione dell'operazione. C'è una funzione UnDo, sotto il menu Edit, per quando si fanno errori più importanti dello sbagliare la scelta di un tool.

Forniamo adesso un quadro riassuntivo dei tool disponibili:

sotto la lista dei tool ci sono 2 colonne per le funzioni. La colonna più a sinistra consiste appunto degli strumenti stessi, mentre quella più a destra, contiene i tool per manipolare oggetti; ovviamente una volta che essi siano già stati disegnati. Segue una lista, con descrizioni dei tool:

Analisi delle principali funzioni

LINE - È la consueta linea: si seleziona per prima cosa l'inizio e poi muovendo il mouse nella direzione e punto voluto, premendo si punta la fine della stessa.

RECTANGLE - È simile alla linea, soltanto che con la prima chiamata si determina la coordinata estrema sinistra del rettangolo, mentre, premendo nuovamente, quella destra.

POLYGON - Questo permette di creare una forma con più facce, tracciando una serie di linee tra i punti scelti, uno alla volta, con il pulsante sinistro del mouse.

FREEHAND - Diversamente dai sistemi in bit-map,l'opzione "mano libera" è l'ultima funzione da usare. Ciò perché usa molta memoria (è generata da milioni di piccoli segmenti di lunghezza totalmente arbitrara).

ARC - Nell'Aegis Draw, colui che lo usa, seleziona il centro dell'arco e i punti iniziali e finali. Altri package, come Pro Design di American Small Business Computer per l'IBM PC, permettono un numero di opzioni diverse per ogni arco. Tutti i tipi di Mac Draw da noi conosciuti non permettono così tanta facilità, ma ovviamente, sono più semplici da usare.

CIRCLE - È il solito generatore di cerchi: si punta il centro e spostandosi con il mouse si determina la lunghezza del raggio, premendo nuovamente si fissa l'immagine.

TESTO - Questo utilizza il tipico font di Amiga e non dà alcuna altra possibilità di scelta. Il testo nell'Aegis Draw è disegnato tramite vettori, e questo è un altro modo per dire che è fornito matematicamente dal programma. Questo perche colui che lo usa, deve essere messo in grado di ingrandire e ruotare le lettere senza alcuna perdita di risoluzione.

DIMENSION - Il tool di dimensionamento fa calcolare e visualizzare automaticamente (all'Aegis Draw) la distanza tra due punti. Questo è utilizzabile come se si stesse disegnando una linea. La distanza in unità è immessa nel disegno tra due frecce che seguono la linea tra l'inizio e la fine dei punti prescelti.

PART - Un ottimo particolare dell'Aegis Draw è la sua possibilità di mantenere un database delle parti usate. Un esempio di questi oggetti, può essere un pezzo di mobile per un disegno architettonico o un componente elettronico o per uno schema.

Il comando Part permette di inserire una parte, precedentemente nominata e definita usando il menu Edit: l'utilità della lista Part è di poter usare le stesse parti della lista in molti disegni.

Altri package hanno simili facilitazioni. Nel A Vision dell'IBM c'è "Templates" - essenzialmente una seconda finestra per disegni. Anche Aegis supporta tutto questo, ma tale comando è abbastanza raro da usare nei campi d'uso tipici del programma. L'Aegis ha comunque promesso di ampliare questa funzione al più presto.

Manipolatori

DRAG IT - Muove un oggetto da una parte all'altra. ROTATE - L'oggetto selezionato può essere ruotato intorno ad un punto definito. Il controllo della rotazione può essere veramente preciso. In confronto, A Vision può muoversi di soli 90 gradi.

CLONE - Riproduce un disegno.

ERASER - Cancella un disegno.

EXPLODE - Per chi non è ancora avvezzo ai termini della grafica object-orien-

ted, questo comando può suonare strano. Normalmente, un disegno può essere formato dall'insieme di diversi altri disegni. Il manipolatore Explode cambia, scombussolando il singolo disegno in altri disegni più piccoli.

SIZER - Permette di variare le dimensioni di un disegno.

BACK - A causa del fatto che l'Aegis Draw tratta i disegni come entità individuali, è possibile per una figura trovarsi sovrapposta ad un'altra. Il manipolatore back muove, in avanti un disegno selezionato dietro ad un altro, dando la possibilità di scegliere l'ordine di priorità.

COLOR - Questo comando permette molto di più che il semplice cambio di colore di un oggetto. Aegis Draw permette agli oggetti di essere di colori differenti. Inoltre, rende le linee più sottili.

STATS - Se si vuole che un cerchio sia centrato in (220,327), questo comando permetterà di immettere questie coordinate in modo diretto. Più precisamente, quello che si può cambiare dipende dal disegno che che si sta utilizzando. Test, per esempio, permette di cambiare la posizione, l'altezza , la larghezza e il testo stesso.

Preference...

Accanto agli attuali disegni, con i tool di AegisDraw se ne possono creare e manipolare altri : oltre a questa possibilità ci sono anche un certo numero di opzioni ridefinibili dall'utilizzatore. Per esempio:

RULERS - È possibile attivarlo o disattivarlo, o selezionare l'unità di misura: tipo Metric o Imperial.

GRID SNAP - Quando questo comando è selezionato, tutte le operazioni di disegno sono forzate a cominciare e a finire sui bordi della griglia

SMOOTHING - Questo comando favorisce l'ammorbidirsi delle linee troppo dritte. Per esempio una serie di denti di sega diverrà una sinusoide.

Aegis Draw ha una capacità multipla di livelli, circa 250 per diagramma, ciò permette di rendere invisibili certi livelli e di evindenziarne altri. Oppure mostrare, o permettere il trattamento, solo di alcune parti di un disegno; proteggendone il resto.

Questo e abbastanza utile per apportare un numero di revisioni su un certo disegno. Si può ritornare ad una particolare revisione solamente disattivando alcuni livelli.

Il package, inoltre, ha la capacità di supportare diverse immagini, contemporaneamente, in memoria Con 512k si è limitati a circa due. Se si hanno altri programmi in esecuzione in background, i limiti aumenteranno ulteriormente.

In fine, Aegis Draw è capace di produrre file di disegni in standard IFF. Possono pertanto editare disegni creati con Draw in Deluxe Paint, Aegis Images, o in ogni altro programma che possieda lo standard IFF.

Naturalmente ci sono milioni di altre opzioni e dettagi: questo package e un po' comprensivo di tutto, e per descriverlo a fondo ci servirebbe un libro.

Niente è perfetto

Ciò che maggiormente ci è parso lacunoso in AegisDraw, sono le sue opzioni di output. Per prima cosa ci permettiamo di encomiare la Aegis per lo sviluppo di un intelligente sistema di driver per plotter che permette all'utente di riscrivere i propri drive. Con la possibilità di utilizzare tale sistema, sembra quasi criminale includere solo un driver già realizzato: quello per il Roland DXY-980.

Questo driver supporta una grafica standard chiamata HPGL (Hewlett Packard Graphics Language) ma anche così, sembra quasi una presa in giro piuttosto di un utile aiuto per l'utente; inoltre include driver completati per pochi altri plotter.

Persino peggio, secondo noi, è che non esiste nessuna ragionabile opzione per un output dedicato a stampanti d'alta qualità (non plotter).

L'unico output su stampante esistente, non è altro che un dump di schermo; il quale ci mostrerà pure i menu del programma. Come se non bastasse ,la stampa così ottenibile appare nella tipica, spaventosa risoluzione grafica caratteristica di un package in bit-map. Pertanto l'utilizzatore medio non potrà avere accesso alle migliori possibilità di questo programma. Noi avremmo gradito per esempio, un output di qualità comparabile al Pro Design II.

Come informazione finale, il manuale afferma che esiste un'opzione per aggiustare l'altezza e la larghezza, ma il foglio che vi è incluso dice chiaramente che quest'opzione non esiste più.

Su questo foglio, l'Aegis afferma che la capacità di selezionare l'area di output è stata cosiderata ridondante poiché l'informazione era gia stata fornita nel driver del plotter.

Questo significa che se volessimo creare un disegno di 8.5 x 11" su di un plotter da 11 x 17", dovremmo riscrivere il plotter driver! Assurdo!

Per concludere, e questo è un rilievo di minor peso, AegisDraw potrebbe essere un po' più veloce. MacDraft è molto più veloce, sebbene non siano stati fatti dei test di comparazione molto approfonditi: non ne abbiamo sentito la necessità.

Tutto sommato questo visto è un buon package soprattutto per quei grafici che possiedano una propra stampante di tipo plotter. Fintanto la Aegis produrra dei veri driver per stampanti, che possano mettere in rilievo la qualità dei disegni ottenibili, è difficile raccomandare questo prodotto a chi sia sprovvisto, appunto, di un proprio plotter.

Il prezzo e ragionevole, e il manuale è di buona qualità. Inoltre, AegisDraw non è protetto, ed e quindi possibile farne delle copie di lavoro.

Conclusioni e precisazioni

Mentre stavamo completando il presente articolo, ci è giunta in redazione l'ultima versione di AegisDraw e quindi diamo in breve notizia delle nuove opzioni in essa contenute.

La presenza di un "Pick Plotter Requester" fa variare in parte la prospettiva da noi più sopra tracciata, infatti, per suo mezzo ci sarà possibile scegliere tra un buon numero di driver per plotter individuabili sul disco di sistema. La Aegis ne ha aggiunti circa cinque nuovi driver in questo drawer. Si ha inoltre modo di credere che fra non molto verranno commercializzati dei driver per stampanti ad alta qualità; quando, però, non lo si sa ancora.

Piedi, pollici, opzioni di frazionamento, e righelli, permetterenno incrementi inferiori di 1/128 di pollice (i calcoli nel sistema metrico decimale li potete fare da soli).

Un display migliorato nella leggibilità ci consente di avere sempre sotto controllo angoli di rotazione, altezze e larghezze di rettangoli, e lunghezze di segmenti.

In fine la Aegis ha anche risolto il problema di qualche vecchio bug: in precedenza, la funzione "smoot", apparentemente, trasformava linee rette in linee ondulate...

Ci troviamo quindi di fronte ad un prodotto complessivamente valido, altri package analoghi hanno prezzi di moto superiori, anche se in qualche sua parte ancora carente. Ciò però non deve essere di freno alcuno per coloro che pensassero di utilizzarlo anche per scopi professionali, infatti, sempre pensando al rapporto qualità prezzo, farebbero senz'altro un buon acquisto.

3D IN BASIC

Introduzione alla grafica tridimensionale

di Alessandro Prandi

Il punto di vista...

Lo scopo principale di questo articolo e del programma ad esso annesso, è senz'altro quello di farvi avvicinare alla grafica in 3D senza particolari difficoltà. Detto questo, dobbiamo subito chiarire un concetto di base, iniziamo con il domandarci: "Come posso disegnare un'immagine a 3 dimensioni su uno schermo a due dimensioni?". Il primo concetto che dobbiamo fare nostro riguarda il piano di proiezione. Un esempio molto semplice: fate conto di guardare il palo della luce fuori dalla vostra finestra, a questo punto segnate sul vetro la cima e la base del palo, così come lo vedete, ora se congiungete i due punti che avete marcato avrete proiettato il palo della luce su un piano di proiezione, la finestra.

La misura del palo della luce che avete riportato sulla finestra dipende da 3 cose: dall'altezza del palo, dalla sua distanza, e da quanto il vostro occhio è distante dalla finestra.

Per rappresentare un punto in uno spazio tridimensionale potete usare i valori X, Y e Z, dove X è la distanza orizzontale, Y la distanza verticale, e Z la profondità. Il palo della luce, la finestra e il vostro occhio sono rappresentati nella Figura 1 con questi parametri.

Se il palo della luce è alto Y l'immagine che avete disegnato sulla finestra sarà il risultato di un computo che chiameremo YW. La distanza che intercorre tra il vostro occhio e la finestra invece la chiamiamo DW. Se osservate la Figura 1 noterete due triangoli: uno grande che va dal punto in cui si trova l'occhio fino al palo della luce, ed uno più piccolo che parte sempre dall'occhio ma che va solo sino all'immagine disegnata sulla finestra. Poiché questi due triangoli sono simili il rapporto tra i loro lati sarà uguale. Questa affermazione ci consente di scrivere:

$$YW = \frac{Y}{Z} \times DW$$

Se invece guardate ad un pezzo di legno che si trova per terra e disegnate la sua immagine sempre sulla finestra, succede esattamente la stessa cosa, solo in direzione orizzontale come mostra la Figura 2.

Se il pezzo di legno è lungo X e l'immagine disegnata sulla finestra è lunga XW ci troviamo nuovamente di fronte a due triangoli e potremo quindi scrivere:

$$\frac{XW}{DW} = \frac{X}{Z} \text{ per trovare XW:}$$

$$XW = \frac{X}{Z} \times DW$$

Se guardate una scatola che si trova nel vostro giardino, ne segnate gli angoli sulla finestra e poi li collegate tra loro con delle linee, otterrete la proiezione di una figura tridimensionale, nello stesso modo come avete fatto per il palo della luce e il pezzo di legno. Ora pensate alla finestra come se fosse lo schermo del vostro monitor. Per definire un'immagine come una serie di punti in uno spazio a 3D collegati tra loro da delle linee, voi potete usare le equazioni XW e YW per calcolare i valori X e Y sullo schermo in modo che corrispondano a ciascun punto dell'immagine. Se collegate i punti calcolati otterrete sul vostro monitor una figura a tre dimensioni.

Prima di poter prendere in considerazione la rotazione di una figura in 3D dobbiamo tener presente che la potremmo vedere da differenti angolazioni, perciò sarà necessario aggiustare un attimo le equazioni XW e YW. Di solito si desidera far ruotare un'oggetto attorno al suo centro. Ed è molto più facile sviluppare l'equazione della rotazione se il centro dell'oggetto si trova al punto 0,0,0.

Nei diagrammi visti precedentemente si presumeva che l'occhio fosse a 0,0,0 e che l'oggetto fosse ad una certa distanza (Z) dal nostro occhio. Se definiamo un'immagine attorno al punto 0,0,0 e l'occhio si trova anche in questo punto allora avremo l'impressione di essere all'interno dell'oggetto. Per rimediare a questa situazione dobbiamo spostare il punto 0,0,0 da dov'è il nostro occhio alla locazione in cui noi vogliamo che la figura ruoti su se stessa. Se scegliamo la varibile DI come la distanza che ci separa dall'immagine in rotazione allora rappresenteremo il triangolo più grande come nella Figura 3.

Quello che prima era semplicemente il valore Z ora diventa DI+Z per cui l'equazione sarà:

$$XW = \frac{X}{Z+DI} \times DW \text{ e}$$

$$YW = \frac{Y}{Z+DI} \times DW$$

Poiché DI è la distanza tra l'occhio e il centro dell'immagine le equazioni appena viste ci permettono di variare la distanza della figura rispetto all'occhio intervenendo su DI. Usando le equazioni YW e XW qualsiasi punto in uno spazio a 3D può

essere proiettato sulla nostra finestra bidimensionale come mostra la Figura 4.

Dalle equazioni al programma

Ora dovremo perfezionare le equazioni per poterle usare sull'Amiga. Poiché le differenze di risoluzione verticale e orizzontale dello schermo richiedono l'uso di un coefficiente, per ovviare all'inconveniente si moltiplica il valore di X per questo coefficiente che chiamiamo SF. Dato che Y si incrementa man mano che vi spostate in basso sullo schermo, invertiremo il segno del valore Y. Infine per poter muovere l'immagine sullo schermo aggiungiamo i valori PX e PY ai valori computati X e Y, per cui avremo:

$$XW = PX + \left(\frac{X}{Z+DI} \times DW \right) \times SF$$

$$YW = PY - \left(\frac{Y}{Z+DI} \times DW \right)$$

Troverete queste equazioni nel programma che accompagna questo articolo. I valori che si addicono maggiormente all'Amiga per quanto riguarda l'alta risoluzione possono essere riassunti in 2.35 per SF, 900 per DI (distanza) e 400 per DW, perlomeno per le immagini che troverete nel programma. (Se cambiate la risoluzione dello schermo dovete modificare SF di conseguenza.)

Figura 1

Per esempio facciamo conto di voler posizionare una figura a PX=200 e PY=150 sullo schermo. Se uno dei punti della figura ha i valori di X=80, Y=90 e Z=100 il punto apparirà sullo schermo a X=275 e Y=114 come segue:

$$XW = PX + \left(\frac{X}{Z+DI} \times DW \right) \times SF \text{ sta per:}$$

$$200 + \left(\frac{80}{100+900} \times 400 \right) \times 2.35 = 275$$

$$YW = PY - \left(\frac{Y}{Z+DI} \times DW \right) \text{ sta per:}$$

$$150 - \left(\frac{90}{100+900} \times 400 \right) = 114$$

Ora consideriamo come ruotare l'immagine da diverse angolazioni (se volete saltare la parte matematica potete continuare la lettura dell'articolo dal paragrafo che inizia con "Le risultanti equazioni che ne derivano sono...")

Gira... gira...

Immaginiamo di avere una linea che attraversi lo schermo da parte a parte in senso verticale e che passi allo stesso tempo per il centro della figura, questa linea la chiamiamo asse Y. Allo stesso modo immaginiamo un linea che attraversi lo schermo orizzontalmente e che passi sempre per il centro della figura e la chiamiamo asse X.

L'asse Z sarà, di conseguenza, una linea che partendo dal nostro occhio attraverserà il centro dell'immagine. Disegniamo un punto alla destra dell'asse X e quindi lo ruotiamo di 30 gradi attorno all'intersezione degli assi X e Y. Quale sarà la locazione di questo punto?

Come illustrato nella Figura 5 il valore del nuovo punto XR è inferiore al valore di X poiché il punto non è lontano sull'asse X, ed anche Y non è inferiore a 0 per cui YR avrà un valore positivo. Poiché la distanza tra la nuova locazione e l'origine (per origine si intende il punto d'intersezione tra le assi X e Y) rimane la stessa durante la rotazione, la linea dall'origine al nuovo punto (XR, YR) sarà della stessa lunghezza del valore originario di X.

La linea che congiunge il punto origine con il nuovo punto (XR, YR) è l'ipotenusa di un triangolo rettangolo, mentre la linea che congiunge il punto origine con il punto disegnato sull'asse X è la base del triangolo. Il COSENO di un angolo viene dato dal lato adiacente dello stesso diviso l'ipotenusa, così:

Figura 2

COS (30) = XR/X oppure XR = X * COS (30)

E poiche il SENO di un un angolo è dato dal lato opposto all'angolo diviso per l'ipotenusa, allora avremo:

SIN (30) = YR/X oppure YR = X * SIN (30)

Come avremmo fatto se il punto, invece di disegnarlo sull'asse X l'avessimo disegnato sull'asse Y? Osserviamo la figura 6, vediamo questa volta che la distanza dal nuovo punto (XR, YR) al punto origine è della stessa lunghezza del valore originale di Y.

Il nuovo valore di X (XR) in questo caso sarà negativo è può essere trovato con:

SIN (30) = -XR/Y oppure XR = -Y * SIN (30)

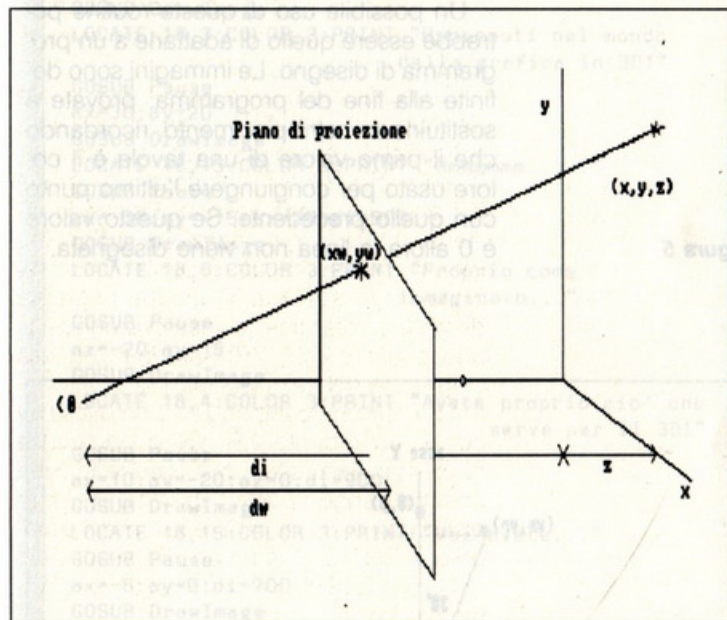
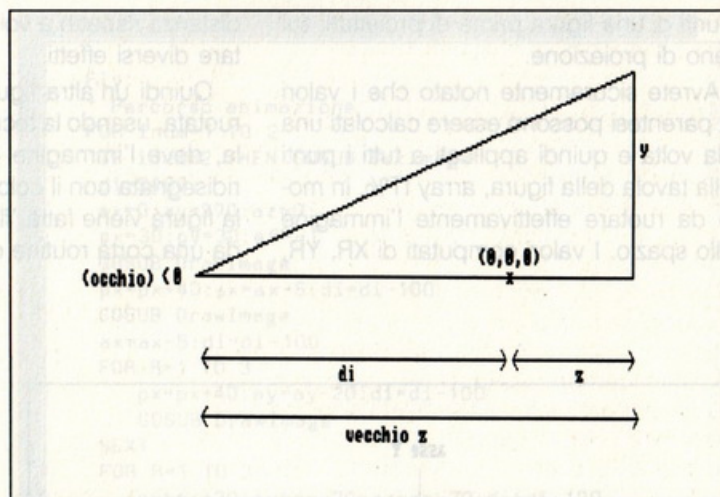



Figura 3

Figura 4

Il nuovo valore Y (YR) può essere ricavato con:

COS (30) = YR/Y oppure YR = Y * COS (30)

Combinando il punto sull'asse X e il punto sull'asse Y nei vari casi otteremo:

XR = X * COS (30) - Y * SIN (30)
YR = X * SIN (30) + Y * COS (30)
ZR = Z

Le equazioni riportate qui sopra possono essere applicate a qualsiasi punto nell'ordine di rotazione di 30 gradi sull'asse Z. L'equazione ZR è stata inclusa per mostrare come il valore Z non cambia, poiché il punto ruota proprio su quest'asse.

Se usiamo le variabili SZ = SIN (Y-angolo) e CZ = COS (Z-angolo) dove 'Z-angolo' sta per quanto vogliamo ruotare l'immagine sull'asse Z, allora le equazioni diventano:

XR = X * CZ - Y * SZ
(A) YR = X * SZ + Y * CZ
ZR = Z

Come faremo invece per ruotare un punto attorno all'asse Y? Useremo SY = SIN (Y-angolo) e CY = COS (Y-angolo) dove 'Y-angolo' sta per quanto vogliamo ruotare l'immagine sull'asse Y, e con un procedimento similare le equazioni diventano:

XR = X * CY - Z * SY
(B) ZR = X * SY + Z * CY
YR = Y

Come faremo per ruotare un punto attorno all'asse X? Di nuovo usando SX = SIN (X-angolo) e CX = COS (X-angolo) dove 'X-angolo' sta per quanto vogliamo ruotare l'immagine sull'asse X, e le equazioni diventano:

YR = Y * CX - Z * SX
(C) ZR = Y * SX + Z * CX
XR = X

Ora è la volta delle sostituzioni. Se prendiamo i valori XR, YR e ZR dalle equazioni in (A) e gli sostituiamo per i valori X,Y e Z delle equazioni in (B) e a sua volta prendiamo i valori XR, YR e ZR che ne risultano, sempre in (B), e sostituiamo ai valori X, Y e Z delle equazioni in (C), otterremo una serie di equazioni che ci consentiranno la rotazione simultanea attorno a ciascun asse.

(Al posto della sostituzione potete adottare un altro sistema ovvero: convertire le equazioni in (A), (B) e (C) in forma di matrice e moltiplicarle insieme per ottenere lo stesso risultato.)

Le risultanti equazioni che ne derivano sono:

XR = X*(CY*CZ) + Y*(-CY*SZ) + Z*(-SY)
YR = X*(CX*SZ-SX*SY*CZ) + Y*(CX*CZ + SX*SY*SZ) + Z*(-SX*CY)
ZR = X*(SX*SZ + CX*SY*CZ) + Y*(SX*CZ-CX*SY*SZ) + Z*(CX*CY)

I programmi

Queste sono le equazioni usate nel programma 3D, che troverete di seguito all'articolo, per eseguire la rotazione di tutti

i punti di una figura prima di proiettarli sul piano di proiezione.

Avrete sicuramente notato che i valori tra parentesi possono essere calcolati una sola volta e quindi applicati a tutti i punti della tavola della figura, array IT%, in modo da ruotare effettivamente l'immagine nello spazio. I valori computati di XR, YR,

Figura 5

e ZR sono depositati in una tavola di rotazione dell'immagine, chiamata RIM%. Infine le equazioni di proiezione esaminate prima vengono impiegate per 'vedere' l'immagine 3D ruotata sul piano di proiezione (il vostro schermo).

A seconda della sequenza con la quale calcoliamo le equazioni di rotazione, una figura può ruotare sul proprio asse o fuori piano, questo dipende dagli angoli di rotazione. Un ultimo commento riguardo i calcoli per la rotazione è d'obbligo. Normalmente noi pensiamo agli angoli e li colleghiamo ai gradi (angolo retto - 90 gradi), mentre il Basic, diversamente, svolge le funzioni di SENO e COSENO calcolando gli angoli in radianti.

Poiché 2 pigreco radianti = 360 gradi, per convertire i gradi in radianti dovete dividere per 57.2958. Questo calcolo è svolto nel programma dalle funzioni trigonometriche.

Il programma usa le stesse equazioni di proiezione e di rotazione che abbiamo esaminato nell'articolo. Quando lo mandate in esecuzione esso, per prima cosa, vi saluta con un'immagine in 3D che ruota nelle differenti angolazioni X, Y e Z e varia la distanza rispetto a voi in modo da presentare diversi effetti.

Quindi un'altra figura viene mostrata e ruotata, usando la tecnica disegna/cancella, dove l'immagine da cancellare viene ridisegnata con il colore dello sfondo. Ora la figura viene fatta 'fluttuare' nello spazio da una corta routine che la muove anche

Figura 6

verso di voi, manipolando la variabile DI.

A questo punto un'altra immagine attraverserà lo schermo 'volando' e, terminato il volo, il programma vi offrirà un menù selezionabile da mouse. Le varie opzioni si possono selezionare clickando il tasto destro, mentre il tasto sinistro serve a ripetere l'ultima opzione scelta dal menù.

Le immagini che vengono visualizzate vengono anche chiamate 'gabbie metalliché in quanto tutti gli angoli sono visibili. Infatti vi sembra di vedere attraverso l'oggetto come se aveste i raggi X. Aggiungendo delle routine per la rimozione delle linee nascoste, si potrebbe ottenere la visualizzazione delle sole facciate visibili normalmente. Comunque questo tipo di routine puo' diventare alquanto complessa e va oltre gli scopi di questa introduzione al 3D.

La variabile EF viene usata nel programma come Erase Flag. Se EF=0, l'immagine viene disegnata con il colore trovato nella tavola della figura. Se EF=1, viene disegnata con il colore di sfondo, cioè viene cancellata. Con EF=-1 lo schermo viene ripulito dòpo la rotazione della figura e prima che essa venga ridisegnata.

Un possibile uso di queste routine potrebbe essere quello di adattarle a un programma di disegno. Le immagini sono definite alla fine del programma, provate a sostituirle a vostro piacimento, ricordando che il primo valore di una tavola è il colore usato per congiungere l'ultimo punto con quello precedente. Se questo valore è 0 allora la linea non viene disegnata.

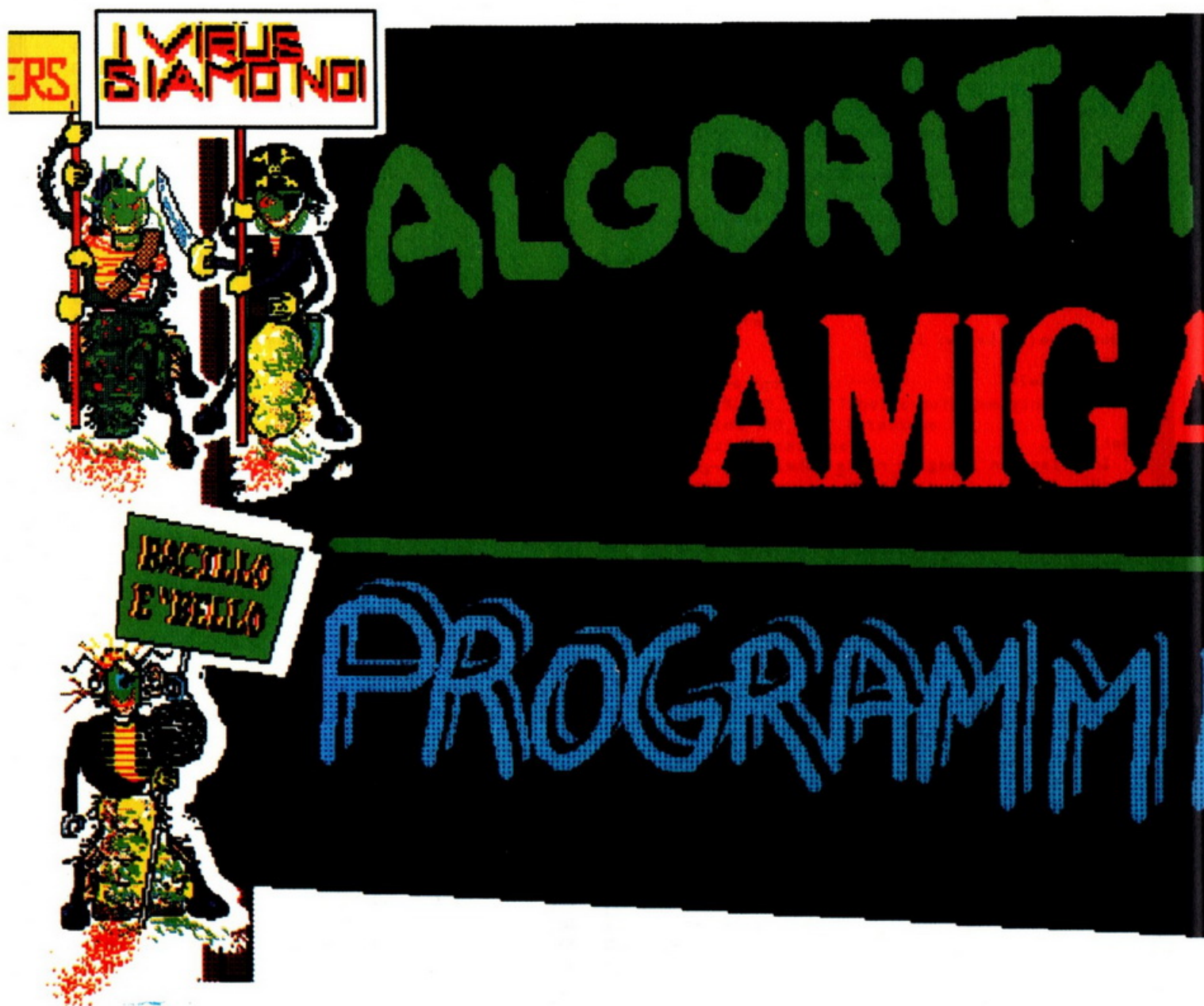
I rimanenti tre valori sono X, Y e Z, valori del punto. Per segnare la fine della vostra figura usate -1 per il colore dell'ultima immissione. Quando definite i punti dell'immagine ricordate che l'oggetto ruoterà attorno al punto 0,0,0. Non ci resta che augurarvi buona fortuna per questo vostro viaggio nel magnifico mondo di 3D.

```
' Prova_3D un esempio di programma di immagini 3D
' Implementazione in AmigaBasic
' di Amiga Magazine Aprile 1988

' Saluti in 3D
CLS:PRINT
LOCATE 10,14:COLOR 3:PRINT " Prova-3D";
COLOR 1:PRINT " -- Un esempio di programmazione in
                3D"
LOCATE 11,36:COLOR 1:PRINT " by Amiga Magazine"
GOSUB InitVals
GOSUB SetImage
' gosub SetImage
' goto Manual
ef=-1
ax=10:ay=5:px=180:py=70
GOSUB DrawImage
LOCATE 18,20:COLOR 3:PRINT"Salve!"
GOSUB Pause
ax=10:ay=-30
GOSUB DrawImage
LOCATE 18,3:COLOR 3:PRINT "Benvenuti nel mondo
                        della grafica in 3D!"
GOSUB Pause
ax=30:ay=20
GOSUB DrawImage
LOCATE 18,15:COLOR 3:PRINT "Hmmmmmm....."
GOSUB Pause
ax=-30:ay=-15:az=20:di=500
GOSUB DrawImage
LOCATE 18,8:COLOR 3:PRINT "Proprio come
                         immaginavo..."
GOSUB Pause
az=-20:ay=15
GOSUB DrawImage
LOCATE 18,4:COLOR 3:PRINT "Avete proprio cio' che
                          serve per il 3D!"
GOSUB Pause
ax=10:ay=-20:az=0:di=900
GOSUB DrawImage
LOCATE 18,15:COLOR 3:PRINT "Voi avete..."
GOSUB Pause
ax=-5:ay=0:di=700
GOSUB DrawImage
LOCATE 18,15:COLOR 3:PRINT "... un AMIGA!!"
FOR i=1 TO 10000:NEXT
ef=0

Ri:
' Immagine rotante
CLS
LOCATE 19,13:COLOR 3:PRINT "Delta Wing Fighter"
GOSUB SetImage
' Disegna e Cancella l'immagine rotante
ax=-90:ay=270:az=0:px=160:py=100
FOR ii=1 TO 4
 GOSUB DrawImage
 GOSUB Pause
 ef=1
 GOSUB DrawImage
 ef=0
 ax=-20
 ay=ay-60
 IF ii=1 THEN ax=0:ay=270:az=0
NEXT
```

```
Fly:
' Percorso animazione
FOR inum=1 TO 2
 IF inum=2 THEN GOSUB SetImage
 di=2400
 ax=0:ay=270:az=0
 px=30:py=30:ef=-1
 GOSUB DrawImage
 px=px+40:ax=ax-5:di=di-100
 GOSUB DrawImage
 ax=ax-5:di=di-100
 FOR R=1 TO 3
   px=px+40:ay=ay-20:di=di-100
   GOSUB DrawImage
 NEXT
 FOR R=1 TO 3
   px=px+30:ay=ay-20:az=az-20:di=di-100
   GOSUB DrawImage
 NEXT
 FOR R=1 TO 4
   px=px-20:py=py+10:az=az+10:di=di-80
   GOSUB DrawImage
 NEXT
 FOR R=1 TO 8
   px=px-9*R:py=py+10:az=az+5:di=di-60:ax=ax-5
   GOSUB DrawImage
 NEXT
NEXT

' Consente il controllo manuale
GOSUB SetImage

Manual:

MENU 1,0,1,"Rotazione +"
MENU 1,1,1,"Sull'asse X"
MENU 1,2,1,"Sull'asse Y"
MENU 1,3,1,"Sull'asse Z"

MENU 2,0,1,"Rotazione -"
MENU 2,1,1,"Sull'asse X"
MENU 2,2,1,"Sull'asse Y"
MENU 2,3,1,"Sull'asse Z"

MENU 3,0,1,"Movimenti"
MENU 3,1,1,"Fuori    "
MENU 3,2,1,"Dentro   "
MENU 3,3,1,"Destra   "
MENU 3,4,1,"Sinistra"
MENU 3,5,1,"Su       "
MENU 3,6,1,"Giu'     "

MENU 4,0,1,"Reset    "
MENU 4,1,1,"Angoli   "
MENU 4,2,1,"Distanza "
MENU 4,3,1,"Posizione"
MENU 4,4,1,"Quit     "

ON MENU GOSUB Menus
ON MOUSE GOSUB Mous

m1=1:GOSUB Reeset
m1=2:GOSUB Reeset
m1=3:GOSUB Reeset

MOUSE ON
MENU ON
```

```
Loop:
 IF act=0 THEN inc=1: GOTO Loop
 GOSUB DrawImage
 GOSUB Vals
 IF MOUSE(0)<>-1 THEN act=0 ELSE GOSUB Mous
 GOTO Loop

'-------------
' Subroutines
'-------------
Vals:
 COLOR 1
 LOCATE 1,1:PRINT "Ax, Ay, Az: "ax", "ay", "az
 LOCATE 2,1:PRINT "Px, Py    : "px", "py
 LOCATE 3,1:PRINT "Di         : "di
 COLOR 3
 LOCATE 4,1:PRINT "Usa i Menu per cambiare
                          l'immagine"
 COLOR 2
 LOCATE 5,1:PRINT "(premi il tasto sinistro per
                          ripetere)"
RETURN

Menus:
 act=1
 inc=1
 m0=MENU(0)
 m1=MENU(1)
 ON m0 GOSUB RotateP,RotateM,MoveI,Reeset
RETURN

Mous:
 act=1
 inc=inc+.5
 ON m0 GOSUB RotateP,RotateM,MoveI,Reeset
RETURN

RotateP:
 IF m1=1 THEN ax=ax+10*inc
 IF m1=2 THEN ay=ay+10*inc
 IF m1=3 THEN az=az+10*inc
RETURN

RotateM:
 IF m1=1 THEN ax=ax-10*inc
 IF m1=2 THEÑ ay=ay-10*inc
 IF m1=3 THEN az=az-10*inc
RETURN

MoveI:
 IF m1=1 THEN di=di-50*inc
 IF m1=2 THEN di=di+50*inc
 IF m1=3 THEN px=px+20*inc
 IF m1=4 THEN px=px-20*inc
 IF m1=5 THEN py=py-10*inc
 IF m1=6 THEN py=py+10*inc
RETURN

Reeset:
 IF m1=1 THEN ax=-15:ay=-25:az=0
 IF m1=2 THEN di=1200
 IF m1=3 THEN px=160:py=100
 IF m1=4 THEN MENU OFF:END
RETURN

Pause:
 FOR i=1 TO 4000:NEXT
RETURN

'-----------------
'     3-D  Routines
'-----------------

' ax, ay, az = rotazione angolo negativa
' di = distanza immagine
' dw = distanza window (piano di proiezione)
' px, py = posizine immagine sullo schermo
' sf = fattore di scala
' ef = flag di cancellazione (1=cancella,
'  0=disegna,
'      -1=CLS & disegna)
' I data delle immagini si trovano in fondo
' al programma

InitVals:
 ' Define Arrays
 DIM it%(100,3):' Tabella dell'immagine
 DIM rim%(100,3):' Rotazione
 ' Inizializzazione
 x=0:y=0:z=0
 dw=400:' Distanza Window
 di=900:' Distanza Immagine
 sf=2.35:' Fattore di scala
 ax=0:ay=0:az=0:' Angli negativi
 px=200:py=100:' x,y coordinate Immagine
 ef=0:' Flag di cancellazione
 f=57.29578:' Fattore radianti
RETURN

DrawImage:
' Disegna l'Immagine
 GOSUB Rotate
 GOSUB DrawIt
RETURN
Rotate:
 ' Ordina gli angoli
 sx=SIN(ax/f): cx=COS(ax/f)
 sy=SIN(ay/f): cy=COS(ay/f)
 sz=SIN(az/f): cz=COS(az/f)
 ' Calcolo rotazioni
 xRx=cy*cz
 yRx=-cy*sz
 zRx=-sy
 xRy=cx*sz-sx*sy*cz
 yRy=cx*cz+sx*sy*sz
 zRy=-sx*cy
 xRz=sx*sz+cx*sy*cz
 yRz=sx*cz-cx*sy*cz
 zRz=cx*cy
 ' Rotazione
 np=0

 Rotate1:
 ' Punto successivo
 c=it%(np,0):IF c=-1 THEN RETURN
 x=it%(np,1):y=it%(np,2):z=it%(np,3)
 ' Calcolo nuova posizione
 rim%(np,1)=x*xRx+y*yRx+z*zRx
 rim%(np,2)=x*xRy+y*yRy+z*zRy
 rim%(np,3)=x*xRz+y*yRz+z*zRz
 np=np+1
 GOTO Rotate1

DrawIt:
```

```
np=0: IF ef=-1 THEN CLS

DrawIt1:
 ' Controlla fine Tabella
  c=it%(np,0): IF c=-1 THEN RETURN
 ' Keep for dividing bz yero
  IF (rim%(np,3)+di)=0 THEN rim%(np,3)=rim%(np,3)+1
 ' Calcola x & y
  xw=px+(rim%(np,1)/(rim%(np,3)+di))*dw*sf
  yw=py-(rim%(np,2)/(rim%(np,3)+di))*dw
 ' Disegna la prossima linea o va al punto
 ' successivo
  IF c=0 THEN GOTO JustMove
  colr=c:IF ef=1 THEN colr=0
  LINE (lx,ly)-(xw,yw),colr

  JustMove:   lx=xw: ly=yw
  np=np+1
  GOTO DrawIt1


SetImage:
 ' Inserisce un'immagine nella Tabella
  n=0
  Itloop:
  READ it%(n,0)
  IF it%(n,0)=-1 THEN RETURN
  READ it%(n,1), it%(n,2), it%(n,3)
  n=n+1:GOTO Itloop
 ' Immagine del saluto
 ' Formato DATA dell'Immagine: c,x,y,z
 ' (c=colore, se =0 muove o disegna)
  DATA 0,-50,30,0
  DATA 1,-55,35,10
  DATA 1,-45,0,0
  DATA 1,-20,-60,-30
  DATA 1,20,-60,-30
  DATA 1,20,-60,-30
  DATA 1,45,0,0
  DATA 1,55,35,10
  DATA 1,50,30,0
  DATA 3,30,80,-30
  DATA 3,-30,80,-30
  DATA 3,-50,30,0
  DATA 0,0,22,-30
  DATA 1,0,-4,-36
  DATA 0,-5,0,-30
  DATA 1,0,-4,-36
  DATA 1,5,0,-30
  DATA 0,-20,30,-25
  DATA 1,-35,25,-17
  DATA 1,-20,20,-25
  DATA 1,-5,25,-21
  DATA 1,-20,30,-25
  DATA 2,-20,20,-25
  DATA 0,20,30,-25
  DATA 1,35,25,-17
  DATA 1,20,20,-25
  DATA 1,5,25,-21
  DATA 1,20,30,-25
  DATA 2,20,20,-25
  DATA 0,-20,-26,-22
  DATA 3,0,-34,-30
  DATA 3,20,-26,-22
  DATA 0,-10,-30,-26
  DATA 3,10,-30,-26
  DATA -1
```

```
 ' Immagine del Delta wing fighter
  DATA 0,0,-20,100
  DATA 1,0,20,-100
  DATA 0,50,-20,-100
  DATA 1,0,-20,100
  DATA 1,-50,-20,-100
  DATA 2,0,20,-100
  DATA 2,50,-20,-100
  DATA 2,-50,-20,-100
  DATA 0,-75,0,-100
  DATA 3,0,0,0
  DATA 3,75,0,-100
  DATA 3,-75,0,-100
  DATA -1

 ' Immagine del caccia
  DATA 0,-25,0,-25
  DATA 1,25,0,-25
  DATA 1,25,0,25
  DATA 1,-25,0,25
  DATA 1,-25,0,-25
  DATA 0,-25,25,25
  DATA 3,-25,-25,25
  DATA 3,-25,-25,-25
  DATA 3,-25,25,-25
  DATA 3,-25,25,25
  DATA 0,25,25,25
  DATA 3,25,-25,25
  DATA 3,25,-25,-25
  DATA 3,25,25,-25
  DATA 3,25,25,25
  DATA 0,0,0,-25
  DATA 2,0,0,50
  DATA 2,0,10,25
  DATA 2,0,0,-25
  DATA -1

 ' Immagine assi X Y Z
  DATA 0,-100,0,0
  DATA 1,100,0,0
  DATA 1,80,-20,0
  DATA 0,100,0,0
  DATA 1,80,20,0
  DATA 0,140,14,0
  DATA 1,170,-16,0
  DATA 0,140,-16,0
  DATA 1,170,14,0
  DATA 0,0,-100,0
  DATA 2,0,100,0
  DATA 2,20,80,0
  DATA 0,0,100,0
  DATA 2,-20,80,0
  DATA 0,0,120,0
  DATA 2,0,134,0
  DATA 2,14,148,0
  DATA 0,0,134,0
  DATA 2,-14,148,0
  DATA 0,0,0,100
  DATA 3,0,0,-100
  DATA 3,0,-20,-80
  DATA 0,0,0,-100
  DATA 3,0,20,-80
  DATA 0,-14,14,-140
  DATA 3,16,14,-140
  DATA 3,-14,-16,-140
  DATA 3,16,-16,-140
  DATA -1
```

di Luigi Manzo e Giovanni Michelon

Iniziamo con questo numero una serie di articoli dedicati alla presentazione di alcuni concetti ed algoritmi basilari dell'analisi numerica.

Il nostro intento, con queste pagine, è quello di mostrare l'utilità di numerosi metodi di calcolo finora ritenuti, ingiustamente a nostro avviso, campo d'azione dei soli specialisti, cercando insieme di non stroncare il lettore volenteroso con una trattazione troppo teorica. Non è certamente nostra intenzione fare un corso di matematica applicata, e spesso e volentieri sacrificheremo il rigore logico e le dimostrazioni dei teoremi a favore dell'illustrazione di applicazioni di tipo più spicciolo e "giocoso". Vorremmo tuttavia offrire a chi legge, anche dalle pagine di una rivista destinata ad un pubblico non specializzato, i mezzi per poter affrontare con un minimo di consapevolezza le problematiche che gli verranno sottoposte, in modo che chiunque abbia interesse ad un utilizzo "intelligente" del proprio computer (e la potenza di calcolo di Amiga è un vero invito a nozze in questo senso!) possa risolvere le questioni che gli si presenteranno innanzi.

A tutti coloro che ci seguiranno su queste pagine promettiamo dunque un avventuroso viaggio tra polinomi e funzioni splines, sistemi dinamici ed epsilon di macchina, serie di Fourier e varie altre amenità di questo tipo; a costoro rivolgiamo sin d'ora l'invito a scriverci per inviarci le loro richieste e domande sugli argomenti trattati.

In questa prima puntata, ci doteremo di alcuni strumenti indispensabili per inoltrarci

nel mondo del calcolo numerico. Precisamente, sfateremo subito il dogma dell'infallibilità computazionale del nostro computer stabilendo un paio di concetti riguardanti la precisione di macchina; quindi inizieremo a presentare una classe di funzioni particolarmente simpatiche ed utili: i polinomi algebrici.

Un semplice esempio

Avete mai provato a far sommare tre numeri al vostro calcolatore? Speriamo bene di si, e probabilmente non avrete mai avuto particolari problemi nell'eseguire questa semplice operazione. Tuttavia, inizializzate, usando Amiga Basic ad esempio, le seguenti variabili (in modo diretto se volete) :

a = 0.23371258 e-4
b = 0.33678429 e+2
c =-0.33677811 e+2

e provate a sommarle; con carta e penna troverete il risultato (esatto) :

6.41371258 e-4

Se chiedete invece al calcolatore di valutare:

(a + b) + c troverete 6.408691 e-4 ; mentre digitando

a + (b + c) avrete in risposta 6.413522 e-4

In altre parole, per il vostro calcolatore non vale la proprietà associativa della somma! Un bug dell'Amiga Basic? No, è un semplice effetto collaterale del modo in cui l'interprete basic (ma il discorso vale in generale per ogni compilatore) gestisce i numeri e le operazioni fra di essi.

Cercheremo ora di comprendere un po' meglio i meccanismi fondamentali in base ai quali è possibile rendersi conto dell'affidabilità dei risultati ottenibili con un qualunque algoritmo di calcolo. La parte che seguirà non risulta di lettura propriamente sollazzevole, ma è necessaria per comprendere termini e concetti che verranno spesso richiamati nel seguito: un po' di pazienza quindi e... avanti!

Rappresentazione di macchina

Un generico numero reale x viene rappresentato dalla macchina mediante una particolare notazione detta floating point, che prevede un numero limitato di cifre significative. In luogo di x perciò la macchina memorizzerà un numero, che indicheremo con fl(x), ricavato dalla rappresentazione binaria di x estraendo da questa le prime t cifre significative, ove t è un prefissato intero che dipende dalle caratteristiche del calcolatore e dell'ambiente di calcolo. Comunque l'approssimazione x = xfl(x) può essere fatta in almeno due modi:

— per troncamento
— per arrotondamento

In entrambi i casi, si può dimostrare che l'errore relativo commesso nell'approssimazione è comunque minore di una quantità fissata, detta epsilon di macchina (vedi figura 1.0). Questa costante è il principale parametro utilizzato per valutare la precisione delle operazioni aritmetiche condotte dalla macchina; per l'Amiga Basic esso vale notoriamente 1 e-7.

Problemi ed algoritmi

Fin qui quindi, nulla di nuovo o di particolarmente trascendentale; il concetto fondamentale di tutto questo discorso è comunque che la stessa memorizzazione di un dato comporta l'introduzione di un errore su esso, e che tale errore è maggiorabile dall'epsilon di macchina; bisogna ora vedere come esso si trasmette ai risultati di elaborazioni condotte su tali dati perturbati di partenza.

A questo proposito risultano molto utili i concetti di problema stabile e di algoritmo ben condizionato. Diamo le seguenti definizioni:

Un problema si dice stabile se piccoli errori sui dati portano a piccoli errori sui risultati.

Un algoritmo si dice ben condizionato se il risultato da esso fornito è la soluzione esatta di un problema perturbato con piccoli errori relativi sui dati.

Risulta essenziale distinguere bene i due concetti; una volta stabilito che un problema, inteso come la corrispondenza che lega un certo insieme di dati numerici in entrata con un altro insieme di dati in uscita, è instabile, qualunque sia l'algoritmo che adottiamo per risolverlo dovremo aspettarci comunque dei risultati poco affidabili. Viceversa, risulterà essenziale evitare algoritmi mal condizionati. Essi, applicati anche a problemi di per sè stabili, non risulteranno affidabili proprio perché piccole alterazioni sui dati si ripercuoteranno grandemente sui risultati.

L'analisi della stabilità di un problema si può fare ricorrendo all'epsilon di macchina per valutare maggiorazioni dell'errore sui dati; il condizionamento di un algoritmo si valuta invece a partire dai risultati che esso offre su un generico problema mediante tecniche cosidette di "analisi all'indietro". Comunque, per algoritmi e problemi appena un po' complessi, l'analisi della stabilità con questi metodi diventa una pia illusione, e allora si adottano metodi più sofisticati (e specifici) di valutazione dell'errore, oppure ci si contenta di valutazioni più grossolane, oppure ancora ci si affida alla Divina Provvidenza, confidando nella bontà del problema studiato.

Scherzi a parte, i pochi concetti visti finora ci permettono perlomeno di dire che le quattro operazioni sono problemi stabili, ad eccezione del caso della somma tra addendi di segno opposto e valore assoluto simile. Vedremo più avanti un esempio di questa situazione; a prescindere da ciò, comunque, nell'esempio da noi proposto di somma di tre numeri esistono almeno due (ovvi) algoritmi utilizzabili: possiamo pensare di sommare prima a e b, e quindi di aggiungere c; oppure sommiamo

prima b e c e al risultato aggiungeremo a. L'analisi all'indietro ci mostra che i due metodi non sono equivalenti, e che l'algoritmo di calcolo più stabile è il seguente :

— ordinare i tre addendi in modo che la somma dei primi due risulti la più piccola possibile in valore assoluto;

aggiungere il terzo addendo alla somma dei primi due.

Ciò rende ragione dei risultati ottenuti nell'esempio riportato sopra; osserviamo inoltre che se date, sempre in ambiente Amiga Basic, in modo diretto a + b + c, otteremo il risultato meno preciso tra i due: evidentemente le routine di Amiga Basic non sono state ottimizzate in questo senso!

Ancora un esempio

Un esercizio che alle scuole superiori capita di risolvere migliaia di volte è la ricerca delle soluzioni di un'equazione di secondo grado scritta nella forma:

x^2 + 2*p*x + q = 0

Bene, considereremo questo problema da un punto di vista numerico.

Nell'ipotesi che esistano, le soluzioni reali sono espresse dalla formula riportata in figura 2.0; si può dimostrare che queste relazioni costituiscono un problema stabile, ad eccezione del caso in cui p è positivo e q è circa uguale a p^2.

L'algoritmo invece che calcola la radice di modulo minimo, rappresentato in figura 2.1, risulta mal condizionato, in particolar modo nel caso in cui -q sia molto minore di p^2. Al contrario, il processo di calcolo

Definiamo errore relativo : $e = \frac{|fl(x) - x|}{|x|}$

Vale la seguente relazione : $e \leq \mu B^{1-t}$, dove
μ=1 se c'e' un troncamento
μ=0.5 se c'e' un arrotondamento
B : base della rappresentazione interna dei numeri
(nella quasi totalita' dei computer B=2)
t : numero di bit riservati alla rappresentazione
delle cifre significative di un numero reale
(tipicamente t=23 o 31 per la singola precisione)

Definiamo epsilon di macchina la quantita' :

$eps \triangleq B^{1-t}$

Risulta sempre :

$e \leq eps$

Fig. 1

del tutto analogo che porta all'altra radice è sempre ben condizionato.

A questo punto, è conveniente applicare questo algoritmo per trovare la radice di modulo massimo, e calcolare l'altra utilizzando il fatto che il prodotto delle radici deve essere pari a q. Come sappiamo, l'algoritmo con cui si calcola il rapporto tra due numeri è ben condizionato e tale sarà perciò l'algoritmo complessivo (che riportiamo nella sua forma definitiva in figura 2.2).

Invitiamo i lettori a rendersi conto direttamente delle differenze segnalate implementando i brevi algoritmi proposti; non aspettatevi variazioni strabilianti: per vedere qualcosa dovrete raggiungere condizioni di calcolo piuttosto estreme. Tuttavia gli esempi proposti hanno solo valore didattico e servono a chiarire i concetti sopra sintetizzati; avremo modo di scontrarci successivamente in modo più tangibile con i problemi provocati dall'errore di macchina.

Fenomeni di cancellazione

Un compito che invece bisogna eseguire piuttosto spesso e che ha una notevole rilevanza pratica è il seguente: si supponga di dover calcolare una funzione del tipo:

y = 180 *x^7* (0.025*x^2 - 25.001)

nel punto x = 10.0002. Avrete senz'altro riconosciuto l'espressione di un polinomio algebrico nella variabile x; lo abbiamo scritto in questa forma per evidenziare un aspetto di instabilità delle operazioni elementari che descriveremo più avanti.

Il risultato esatto di questa espressione è 18.002520151215... Se provate ad impostare questa espressione con l'Amiga Basic e a farvi stampare il risultato, dopo aver inizializzato la variabile x, otterrete un numero che non ha niente a che fare con il risultato dato sopra. Ciò, si badi bene, nonostante i dati abbiano un numero di di cifre significative ben al di sotto del massimo memorizzato dalla macchina: non si può dunque parlare di perturbazione sui dati! Si è verificato invece un fenomeno, detto cancellazione, tipico dell'operazione di differenza tra due numeri positivi (è incriminata quindi l'espressione tra parentesi).

Nel nostro esempio tale differenza non è rigorosamente nulla, ma il primo termine è così prossimo al secondo che, nel momento in cui viene memorizzato nel solito formato di floating point dopo essere stato calcolato, tale differenza finisce con lo scomparire.

0) Radici dell' equazione di secondo grado :

$x=p-\sqrt{p^2-q}$ $y=p+\sqrt{p^2-q}$

Ovviamente vale l' ipotesi $p^2 \geq q$

1) Nell' ipotesi che p>0 e q<0, la radice x di modulo minimo si puo' valutare con il seguente algoritmo :

$s \leftarrow p^2$
$z \leftarrow s-q$
$r \leftarrow \sqrt{s}$
$x \leftarrow p-r$

2) Le radici x e y possono venire convenientemente calcolate con il seguente algoritmo :

$s \leftarrow p^2$
$z \leftarrow s-q$
$r \leftarrow \sqrt{s}$
$y \leftarrow p+r$
$x \leftarrow q/r$

Fig. 2

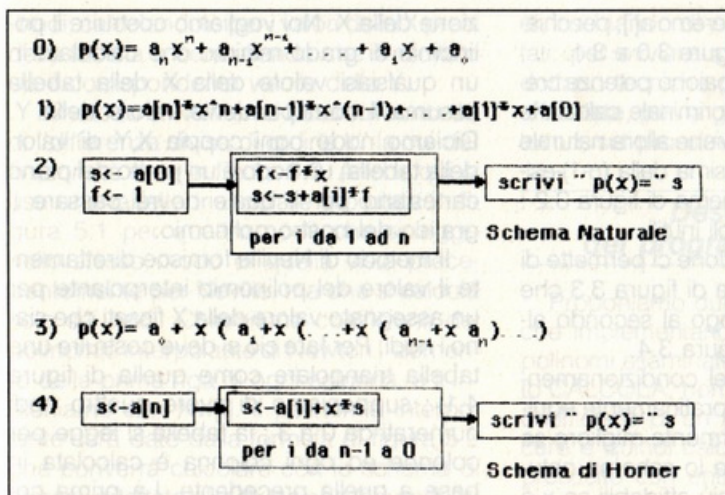



Fig. 3

Quando viene eseguita la differenza, si avrà come risultato 0 (è avvenuta la "cancellazione" appunto di tutte le cifre) e non 1 e-8 come dovrebbe essere. La situazione è inoltre aggravata dal fatto che in realtà le routine di calcolo per qualche misterioso motivo forniscono una serie di cifre pressoché casuale come risultato della differenza, rendendo poi assolutamente inaffidabili le successive elaborazioni su di esso. Sottolineiamo ancora che il grande errore trovato non è da imputare a fenomeni di mal cndizionamento dell'algoritmo di sottrazione, bensì all'instabilità del problema: "sottrazione di numeri quasi uguali", instabilità che tende ad amplificare grandemente le perturbazioni sui dati in ingresso dovute alle elaborazioni precedenti.

In ogni caso, questa evenienza è molto grave e va assolutamente evitata; avremo modo più avanti di segnalarvi casi in cui essa può verificarsi.

I polinomi : introduzione

Nell'esempio precedente abbiamo affermato che i polinomi sono strumenti matematici molto utili nelle applicazioni pratiche : vediamo allora come e quando usarli. Per esempio : vi siete mai chiesti come fa il vostro calcolatore (o come fecero agli inizi dell'Ottocento i matematici, ahimè a mano) a valutare funzioni come sin, cos, exp?

Naturalmente, come avrete già capito, la risposta è : con un polinomio. Non ci credete? Calcolate allora la differenza tra sin(x) ed il valore dato dal seguente polinomio (i più bravi riconosceranno senz'altro il polinomio di Taylor) :

x-x^3/6+x^5/120-x^7/5040+x^9/362880
per -PI/2 < x < PI/2
dove PI = 3.14159265358979323...

Per i più esigenti (!?!) suggeriremo poi un metodo più efficace.

Abbiamo così individuato un primo campo d'applicazione : la valutazione di funzioni che, sebbene rigorosamente definite, non sono poi sempre facili da calcolare ; non solo, potremmo anche pensare di utilizzare i polinomi per approssimare funzioni che altrimenti richiederebbero una notevole mole di calcoli.

Un ulteriore settore d'impiego è suggerito dal seguente problema : disponiamo di una serie di valori, magari ottenuti sperimentalmente dalle misure su una grandezza fisica o direttamente da una tabella, al variare di una qualche incognita; sappiamo che i valori sono legati alla nostra incognita da una funzione che però non conosciamo : vogliamo ciò nonostante calcolare la funzione per valori dell'incognita non presenti in tabella o non direttamente misurati.

Come si fa? Cosa c'entrano i polinomi?

Semplice : si prende un polinomio di grado opportuno e lo si "costringe" ad assumere i valori di cui siamo in possesso; possiamo sperare che il nostro polinomio approssimerà bene (vedremo più avanti quanto bene) la funzione sconosciuta. È questo il cosiddetto problema dell'interpolazione polinomiale.

0)

X	Y
x_0	y_0
x_1	y_1
x_2	y_2
..	..
..	..
x_n	y_n

1) $P_0 = y_0$ → P_{01} → P_{012} → P_{0123}; $P_1 = y_1$ → P_{01}, P_{12}; P_{12} → P_{012}, P_{123}; $P_2 = y_2$ → P_{12}, P_{23}; P_{23} → P_{123}; $P_3 = y_3$ → P_{23}; P_{123} → P_{0123}

2)
$$P_{01} = \frac{P_0\,(x-x_1) - P_1\,(x-x_0)}{x_0 - x_1}$$
$$P_{23} = \frac{P_2\,(x-x_3) - P_3\,(x-x_2)}{x_2 - x_3}$$

3)
$$P_{012} = \frac{P_{01}\,(x-x_2) - P_{12}\,(x-x_0)}{x_0 - x_2}$$
$$P_{123} = \frac{P_{12}\,(x-x_3) - P_{23}\,(x-x_1)}{x_1 - x_3}$$

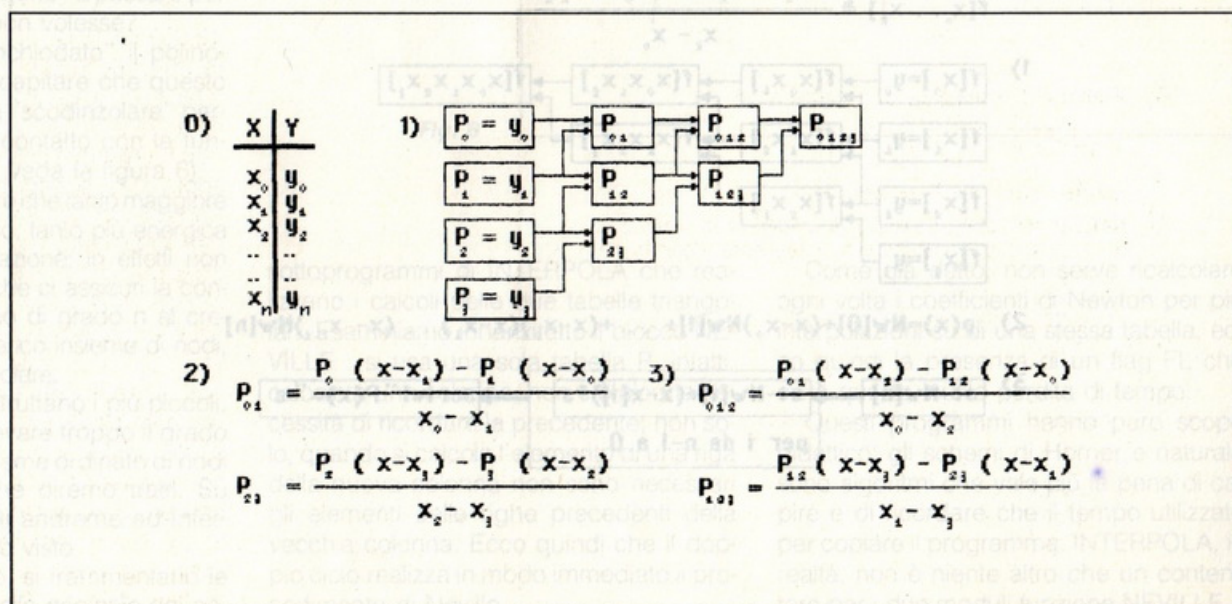



Fig. 4

Ci sembra opportuno far notare che in campo tecnico-scientifico molto spesso si devono risolvere problemi di interpolazione di dati da una tabella; si pensi per esempio alle dilatazioni di un certo materiale per effetto della temperatura o delle forze applicatevi, o all'inviluppo di una forma d'onda campionata; a tutti voi sarà poi capitato di dover utilizzare una termocoppia al platino-rodio per misurare temperature intorno ai 2000 K! Vi sarete senz'altro accorti che le sue caratteristiche d'uscita sono fornite dando i coefficienti di opportuni polinomi.

Utilizzazioni esotiche a parte, esamineremo per il momento i due seguenti problemi:
— noti i coefficienti, come calcolare un polinomio
— noti alcuni valori, come interpolarli mediante un polinomio.

Prima di iniziare a descrivere alcune soluzioni, vogliamo sin d'ora annunciare che tratteremo ancora l'argomento polinomi e loro applicazioni in successivi articoli di questa serie.

Lo schema naturale e quello di Horner

Supponiamo allora i coefficienti del nostro polinomio p(x) noti e memorizzati in una tabella i cui elementi chiameremo a[i]; per chiarezza si confrontino le figure 3.0 e 3.1.

Osserviamo che compaiono potenze crescenti della x : sarebbe criminale calcolarle tutte separatamente! Ci viene allora naturale dedurre la potenza (n)-esima dalla (n-1)-esima, cioè adottare lo schema di figura 3.2 : si risparmiano così calcoli inutili.

Un'ulteriore osservazione ci permette di scrivere p(x) nella forma di figura 3.3 che dà immediatamente luogo al secondo algoritmo di calcolo di figura 3.4.

Dal punto di vista del condizionamento, i due algoritmi sono praticamente equivalenti; Horner è leggermente migliore se x è minore di 1, mentre lo schema naturale dà risultati un po' più affidabili se x è piuttosto grande. Si noti però che con Horner si fanno meno calcoli, e che non compaiono mai potenze troppo elevate della x, evitando così possibili overflow.

Interpolazione : i metodi di Neville e Newton

Il problema dell'interpolazione è decisamente più complesso del precedente; per fissare meglio le idee, diamo un'occhiata alla tabella di figura 4.0; sia Y funzione della X. Noi vogliamo costruire il polinomio di grado minimo che calcolato in un qualsiasi valore della X della tabella assuma il corrispondente valore della Y. Diciamo nodo ogni coppia X,Y di valori della tabella; un nodo è un punto del piano cartesiano per il quale dovrà passare il grafico del nostro polinomio.

Il metodo di Neville fornisce direttamente il valore del polinomio interpolante per un assegnato valore della X fissati che siano i nodi. Per fare ciò si deve costruire una tabella triangolare come quella di figura 4.1 ; supponiamo di avere quattro nodi numerati da 0 a 3 ; la tabella si legge per colonne ed ogni colonna è calcolata in base a quella precedente. La prima colonna rappresenta i valori della Y : li indicheremo con una P ed un indice che va da 0 a 3. Un elemento della seconda colonna è calcolato in base alla formula di figura 4.2 e rappresenta una retta che collega i due nodi segnalati dall'indice. Un elemento della terza colonna si calcola con la formula di figura 4.3 e rappresenta una parabola per tre nodi. Si calcolano così polinomi di grado sempre maggiore fino a giungere al polinomio interpolante che cercavamo. Potete verificare che il polinomio così trovato passa effettivamente per

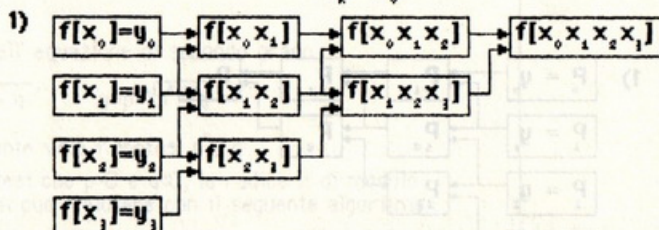
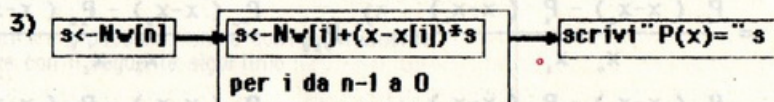



Fig. 5

i nodi, calcolandolo per i valori della X presi dalla tabella e controllando che il risultato sia il corrispondente valore della Y.

Il metodo di Newton si basa sul concetto di "differenza divisa di ordine k" la cui definizione è data in figura 5.0. Anche qui si usa una tabella triangolare (riportata in figura 5.1 per quattro nodi) che si legge nello stesso modo di quella vista precedentemente per Neville, ma che si calcola solo sui nodi. Chiamiamo coefficienti del polinomio interpolante di Newton l'elemento della prima riga di ogni colonna, e li indichiamo con Nw[i]. Il polinomio interpolante sarà dato dalla formula di figura 5.2 che converrà calcolare con lo schema di Horner adattato alla situazione (figura 5.3).

Poiché i coefficienti di Newton vengono calcolati una volta per tutte, questo metodo si rivela più veloce nel caso in cui bisogna eseguire più interpolazioni sulla medesima tabella; complessivamente infatti bisogna eseguire meno conti rispetto al metodo di Neville.

Vi avevamo promesso un modo più efficace per valutare sin(x); avrete capito che basta scegliere dei nodi opportuni e facili da calcolare ed usare uno dei metodi proposti. Vi invitiamo a valutare le differenze utilizzando il programma riportato in calce all'articolo. Vi suggeriamo inoltre di scrivere voi stessi dei programmi che traccino i grafici delle curve e delle loro interpolanti polinomiali.

Fenomeno di Runge - Polinomi a tratti

Introducendo il discorso sull'interpolazione, abbiamo detto di voler prendere un polinomio e di "costringerlo" a passare per i nodi: beh, e se lui non volesse?

Noi abbiamo sì "inchiodato" il polinomio ai nodi, ma può capitare che questo si ribelli ed incominci a "scodinzolare" perdendo sempre più il contatto con la funzione interpolanda (si veda la figura 6).

È forse intuitivo notare che tanto maggiore è il grado del polinomio, tanto più energica può essere la sua reazione; in effetti non esiste alcun teorema che ci assicuri la convergenza del polinomio di grado n al crescere di n, su un generico insieme di nodi, alla funzione da interpolare.

Allora, al solito, si sfruttano i più piccoli: conviene cioè non elevare troppo il grado e dividere il nostro insieme ordinato di nodi in più sottoinsiemi che diremo tratti. Su ognuno di questi tratti andremo ad interpolare con un metodo visto.

Così facendo, però, si frammentano le informazioni della tabella originale dei nodi, rischiando di non ottenere buoni risultati: per ovviare a questo problema si adottano tecniche particolari, che saranno affrontate, per la gioia dei nostri due o tre lettori, sul prossimo numero.

Descrizione dei programmi proposti

Proponiamo due semplici programmini che implementano i quattro algoritmi sui polinomi esaminati sopra. Il primo chiamato CALCOLA, non fa altro che richiedere i coefficienti di un polinomio a vostro piacere e quindi calcolarlo nel punto da voi prescelto con entrambi gli schemi, naturale e di Horner.

Il secondo programma, chiamato INTERPOLA, è analogo al primo, ma richiede il valore dei nodi anziché quello dei coefficienti. Vale la pena di soffermarci sui sottoprogrammi di INTERPOLA che realizzano i calcoli delle due tabelle triangolari. Esaminiamo innanzitutto il blocco NEVILLE : si usa una sola tabella P, infatti, calcolata una colonna, non c'è alcuna necessità di ricordare la precedente; non solo, quando si calcola l'elemento di una riga della nuova colonna non sono necessari gli elementi delle righe precedenti della vecchia colonna. Ecco quindi che il doppio ciclo realizza in modo immediato il procedimento di Neville.

Non così naturale si presenta il blocco COEFF, che dispone i coefficienti di Newton: qui siamo infatti costretti a ricordare gli elementi della prima riga e possibilmente nello stesso ordine per non complicarci la vita quando dovremo usarli per calcolarci i coefficienti del polinomio di Newton.

Osservando la tabella di figura 5.1 , vediamo che non cambia assolutamente nulla se la disegnamo spostando tutte le colonne dalla seconda alla quarta verso il basso in modo che gli elementi della prima riga diventino quelli della diagonale della tabella triangolare. Con questa nuova configurazione i coefficienti di Newton occupano la diagonale: procedendo come con NEVILLE, con una sola tabella siamo in grado di calcolarci i coefficienti cercati, infatti procedendo di colonna in colonna non modifichiamo mai l'elemento diagonale che così rimane memorizzato per quando ci servirà.

Fig. 6

Come già detto, non serve ricalcolare ogni volta i coefficienti di Newton per più interpolazioni su di una stessa tabella, ecco quindi la presenza di un flag FL che evita questa inutile perdita di tempo.

Questi programmi hanno puro scopo didattico: gli schemi di Horner e naturale sono algoritmi che vale più la pena di capire e di ricordare che il tempo utilizzato per copiare il programma. INTERPOLA, in realtà, non è niente altro che un contenitore per i due moduli funzione NEVILLE e

INFORMATICA

NEWTON, i quali invece non sono poi tanto banali, e che vi conviene utilizzare in vostri programmi.

Conclusioni

Forse per qualcuno di voi questo è stato il primo impatto (speriamo piacevole) con i problemi tipici della matematica numerica: vorremmo fosse chiaro sin d'ora che le soluzioni che essa offre non sono sempre valide universalmente ed applicabili comunque. Non ha alcun senso utilizzare dei metodi a scatola chiusa (del tipo: Neville su 200 nodi) confidando magari sull'"intelligenza" del computer!

Con ciò non vogliamo diminuire l'utilità dei metodi numerici (usati oggi in talmente tanti settori che per valutarne i pregi basta solo gurdarsi in giro), ma piuttosto condannare la mentalità di coloro che preferiscono far pensare una macchina al loro posto.

Meditate gente, meditate!

```
REM Programma calcolo

DIM SHARED a(50)

REM Programma principale

Inizio:
 CLS
 c$=" ": n=0
 WHILE ( c$<>"U" )
   PRINT "Coefficiente a(";n;") ? "
   INPUT a(n)
   n=n+1
   INPUT "U per uscire",c$
 WEND
 c$=" ": n=n-1
 WHILE ( c$<>"U" )
   INPUT "Valore di x  ? ",x
   CALL naturale(x)
   CALL horner(x)
   PRINT "Naturale  = "; nat ;" Horner = ";hor
   INPUT "U per uscire ",c$
 WEND
 INPUT " T per terminare ",c$
IF c$<>"T" THEN GOTO Inizio

SUB naturale(x) STATIC
SHARED nat,a,n
 s=a(0)
 f=1
 FOR i=1 TO n
    f=f*x
    s=s+a(i)*f
 NEXT i
 nat=s
END SUB

SUB horner(x) STATIC
SHARED hor,a,n
 s=a(n)
 FOR i=n-1 TO 0 STEP -1
    s=a(i)+x*s
 NEXT i
 hor=s
END SUB

REM programma interpola

DIM SHARED xn(50),yn(50),Nw(50),P(50)

REM programma principale

Inizio:
```

```
 CLS
 c$=" ":n=0:fl=0
 WHILE (c$<>"U")
   PRINT " Nodo ";n;" x ";
   INPUT xn(n)
   INPUT " y ? ",yn(n)
   n=n+1
   INPUT " U per uscire ",c$
 WEND
 n=n-1:c$=" "
 WHILE (c$<>"U")
   INPUT " Valore x ? ",x
   CALL neville(x)
   CALL newton(x)
   PRINT " Neville = ";nev;" Newton = ";nwt
   INPUT " U per uscire ",c$
 WEND
 INPUT " T per terminare ",c$
 IF c$<>"T" THEN GOTO Inizio

SUB neville(x) STATIC
SHARED xn,yn,n,nev
 FOR i=0 TO n
   P(i)=yn(i)
 NEXT i
 FOR k=1 TO n
   FOR i=0 TO n-k
     P(i)=( (x-xn(i+k))*P(i)-(x-xn(i))*P(i+1) )/
                                 ( xn(i)-xn(i+k) )

   NEXT i
 NEXT k
 nev=P(0)
END SUB

SUB coeff STATIC
SHARED xn,yn,Nw,n,fl
 FOR i=0 TO n
  Nw(i)=yn(i)
 NEXT i
 FOR k=1 TO n
   FOR i=n TO k STEP -1
     Nw(i)=( Nw(i)-Nw(i-1) )/( xn(i)-xn(i-k) )
   NEXT i
 NEXT k
 fl=1
        : REM coefficienti calcolati
END SUB

SUB newton(x) STATIC
SHARED xn,Nw,n,fl,nwt
 IF fl=0 THEN CALL coeff
 s=Nw(n)
 FOR k=n-1 TO 0 STEP -1
  s=Nw(k)+(x-xn(k))*s
 NEXT k
 nwt=s
END SUB
```

PER IL TUO COMPUTER

A. Bigiarini - P. Cecioni - M. Ottolini

IL MANUALE DI AMIGA

Rivolto soprattutto ai programmatori, per saperne di più e conoscere meglio i tre modelli di Amiga e le loro ampie possibilità. Poichè vengono presentate le differenze fra i tre modelli disponibili della macchina, il libro risulta utile anche come una funzionale guida all'acquisto.

SOMMARIO

Caratteristiche generali - Grafica - Sprite - Coprocessori - Audio - Interfacciamento - Chip 8520 - Compatibilità IBM - Rom Kemel - Amiga DOS 1.1 e 1.2 - Registri dei Chip Custom - SuperDOS - ARC - SNOOP 1.0.

244 pagine Cod. CZ532 L. 39.000

R. Bonelli - M. Lunelli

AMIGA 500

GUIDA PER UTENTE

Finalmente un testo in grado di racchiudere in un'unica guida tutte le informazioni necessarie agli utenti di Amiga 500, in modo che possano comprendere tutte le possibilità del loro sistema e utilizzarlo al meglio.

SOMMARIO

Uso del mouse - Uso dei menu - Programmi del disco Workbench - Programmi del disco extras - Amiga Dos - Amiga Basic - Il Basic compilato: AC BASIC - Il True Basic.

370 pagine Cod. CC627 L. 55.000

M. England - D. Lawrence

AMIGA HANDBOOK

Un libro per conoscere l'Amiga, il nuovo computer della COMMODORE, al fine di comprendere e sfruttare al massimo tutte le potenzialità di questo sistema considerato da molti rivoluzionario.

SOMMARIO

Uno sguardo all'Amiga - Chip 68000 - Copper co-processor - Playfield e sprite - Blitter - Comunicazioni con il mondo esterno - Nucleo e Exec - Sistema operativo - Workbench e le tecniche di intuition - DOS e Command line interface - Programmi in BASIC.

204 pagine Cod. CC320 L. 35.000

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

GRUPPO EDITORIALE JACKSON
Via Rosellini, 12 - 20124 MILANO

INDICARE CHIARAMENTE CODICI E QUANTITÀ DEI VOLUMI RICHIESTI							
Codice	Q.tà	Codice	Q.tà	Codice	Q.tà	Codice	Q.tà

L. 4.000 per contributo fisso spese di spedizione

MODALITÀ DI PAGAMENTO

☐ Allego assegno n. ______ di L. ______ della Banca ______

☐ Ho effettuato il pagamento di L. ______ a mezzo: ☐ vaglia postale ☐ vaglia telegrafico ☐ versamento sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson SpA Milano e allego fotocopia della ricevuta.

☐ Pagherò al postino l'importo di L. ______ al ricevimento dell'opera.

☐ Richiedo l'emissione della fattura (formula riservata alle aziende) e comunico il numero di Partita IVA ______

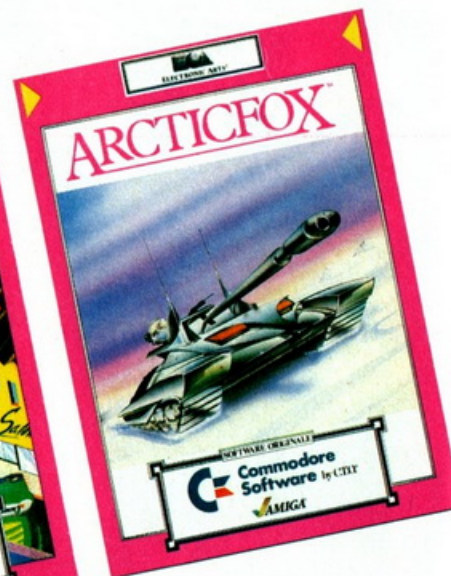
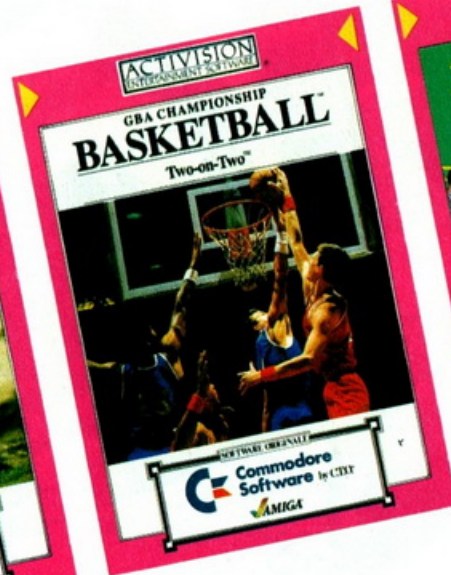
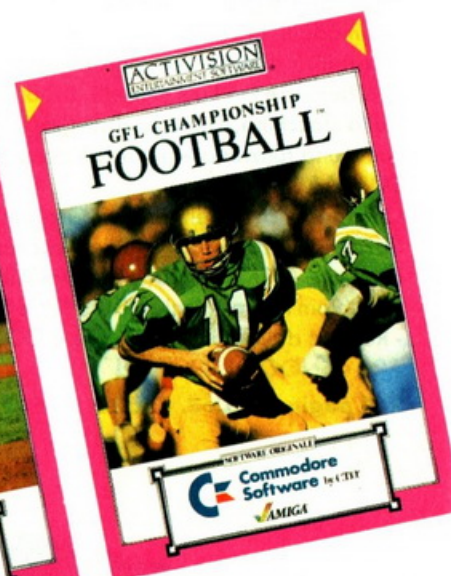
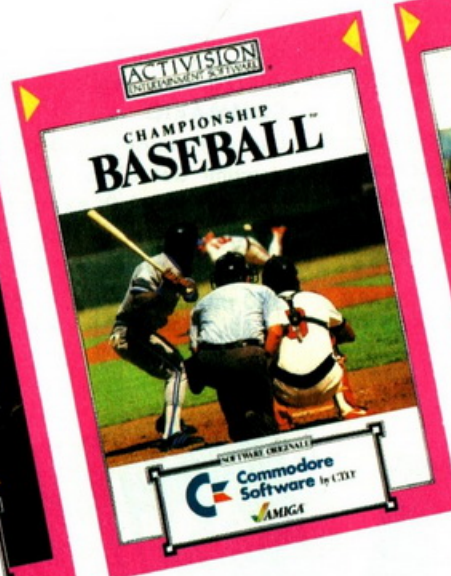
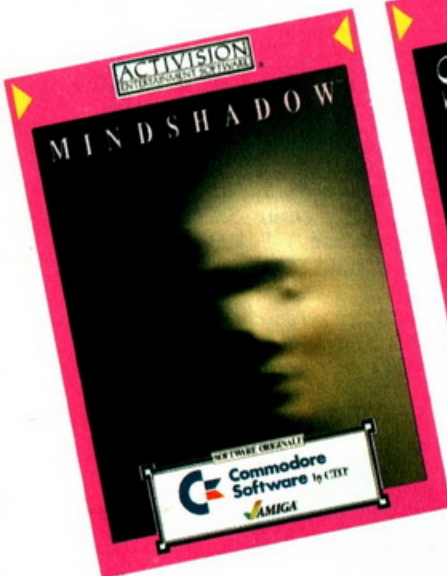
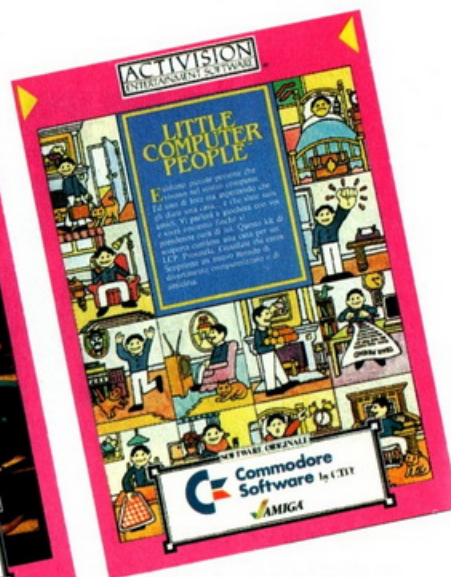
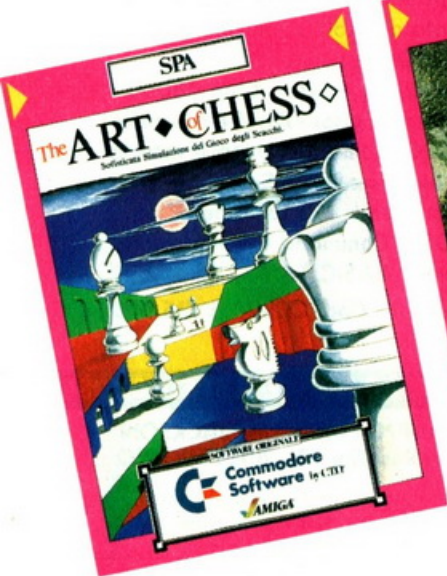
DATA ______ FIRMA ______

COGNOME ______

NOME ______

VIA E NUMERO ______

CITTÀ ______

CAP ______ PROV. ______

C.T.O.

Via Piemonte 7/F
40069 Zola Predosa (BO)
tel. 051/753133 (r.a.)
telefax 051/753418
telex 520659 CTO BO I

Original Software by C.T.O.

Manuali in Italiano

Codice	Titolo	Prezzo di vendita
	ACTIVISION	
ACT101	Hacker II	29.500
ACT103	Shanghai	29.500
ACT104	GBA Championship Golf	29.500
ACT105	GBA Championship Basketball	29.500
ACT106	Championship Baseball	29.500
ACT107	GFL Championship Football	29.500
ACT108	Borrowed Time	33.000
ACT109	Little Computer People	33.000
ACT110	Mindshadow	33.000
ACT111	Tass Times	33.000
ACT112	Portal	48.000
ACT002	Space Quest	35.000 in inglese
	S.P.A.	
SPA101	The Art of Chess	29.500
	ELECTRONIC ARTS	
ECA101	Adventure Construction Set	38.000
ECA102	Artic Fox	29.500
ECA103	Bard's Tale I	29.500
ECA104	Chess Master 2000	29.500
ECA105	Earl Weaver Baseball	29.500
ECA106	Instant Music	33.000
ECA107	Marble Madness	29.500
ECA108	Skyfox	29.500
ECA109	Test Drive	33.000
ECA110	Ferrari Formula One	38.000
ECA601	Art Part I	34.000
ECA602	Art Part II	34.000
ECA603	Hot & Coll Jazz	34.000
ECA604	Rock 'n' Roll	34.000
ECA605	Seasons & Holidays	34.000
ECA701	DELUXE Music Construction Set	94.000
ECA702	DELUXE Paint II	99.000
ECA703	DELUXE Print	90.000
ECA704	DELUXE Video	109.000
ECA011	Black Shadow	33.000 in inglese
	THE DISC COMPANY	
TDC001	Kind Words	60.000
	PROGRESSIVE PERIPHERALS & SOFTWARE	
PPS001	PIXmate	94.000
	TYNESOFT	
TYN001	Winter Olympiad 88	27.000 in inglese
	COMMODORE ITALIANA S.p.A.	
CIT001	Pagesetter	210.000
CIT002	Texteraft Plus	145.000 in inglese
	C.T.O. s.r.l.	
LGX002	Logistix	120.000
SBA004	Superbase Personal	190.000
SBA005	Superbase Professional	399.000 in inglese

BUONO D'ORDINE

Desidero ricevere i seguenti articoli:

CODICE	TITOLO GIOCO	PREZZO
	TOTALE L.	

Cognome ....

Nome ....

Via ....

Cap. .... Città ....

Prov. .... Tel. ....

Firma ....

(se minorenne quella di un genitore)
Gli ordini non firmati non verranno evasi.
Completa le parti del buono d'ordine (o di una sua fotocopia) e spediscilo in busta chiusa a:
AMIGA SOFT SERVICE - Via Rosellini, 12 - 20124 Milano

Per pagamento in contrassegno addebito di L. 6.000 per importi inferiori a L. 35.000; nessun addebito per ordini superiori a L. 35.000.
Pagamento con assegno da allegarsi al coupon d'ordine intestato a : "LENA s.r.l." addebito forfettario di L. 4.000 per importi inferiori a L. 30.000; nessun addebito per ordini superiori a L. 30.000.
Le spedizioni verranno effettuate da "LENA s.r.l." - I prezzi sono intesi al pubblico I.V.A. inclusa.

SERVIZIO LETTORI

IL GRUPPO EDITORIALE JACKSON PROMUOVE OGNI GIORNO NUOVE INIZIATIVE PER FACILITARE IL CONTINUO DIALOGO CON I PROPRI LETTORI. NATURALMENTE È IMPORTANTE CHE QUESTO SCAMBIO DI INFORMAZIONI SIA RESO IL PIÙ POSSIBILE AUTOMATICO E CHE I SUOI TEMPI SIANO SEMPRE PIÙ RISTRETTI. È CON QUESTO INTENTO CHE NASCE IL SERVIZIO LETTORI JACKSON, ORGANIZZATO IN MODO DA SODDISFARE OGNI ESIGENZA, SECONDO UN SISTEMA DI CEDOLE PRECONFIGURATE, DA INVIARE AL NOSTRO SERVIZIO MARKETING. ANZITUTTO, IL SERVIZIO LETTORI JACKSON CONSENTE DI SOTTOSCRIVERE ABBONAMENTI O ORDINARE LIBRI E GRANDI OPERE UTILIZZANDO LE CEDOLE QUI A FIANCO, SCEGLIENDO LA MODALITÀ DI PAGAMENTO PREFERITA. UN ESTRATTO CONDENSATO DEL CATALOGO LIBRI E GRANDI OPERE JACKSON È PUBBLICATO NELLE ULTIME PAGINE DI QUESTA RIVISTA; IL CATALOGO COMPLETO PUÒ ESSERE

COMUNQUE ORDINATO, UTILIZZANDO LA CEDOLA NUMERO 3: INFORMAZIONI & AGGIORNAMENTI. QUEST'ULTIMA È LA PIÙ IMPORTANTE E PERMETTE AL LETTORE DI RICEVERE, DIRETTAMENTE A CASA PROPRIA, TUTTE LE INFORMAZIONI SULLE INIZIATIVE JACKSON CHE LO INTERESSANO: CATALOGHI, LIBRI, CAMPAGNA ABBONAMENTI CORSI DELLA DIVISIONE FORMAZIONE E PRODOTTI PER LA DIDATTICA JACKSON S.A.T.A., COPIE OMAGGIO DI RIVISTE E FASCICOLI DI GRANDI OPERE. QUESTO SERVIZIO CONSENTE, OLTRE CHE DI RIMANERE AGGIORNATI, ANCHE DI AGGIORNARE I COLLEGHI E GLI AMICI, POICHÈ LA CEDOLA È STUDIATA ANCHE CON QUESTO INTENTO. NON PIÙ TELEFONATE E LETTERE: DA OGGI È SUFFICIENTE SPEDIRE L'APPOSITO TAGLIANDO, PER OTTENERE IN BREVISSIMO TEMPO IL MATERIALE DESIDERATO.

CEDOLA ABBONAMENTO RIVISTE JACKSON

Se desiderate sottoscrivere abbonamenti alle riviste Jackson, utilizzate questa cartolina. Gli abbonati Jackson possono contare su un duplice risparmio (una tariffa privilegiata e la garanzia del prezzo bloccato per la durata del proprio abbonamento) e hanno diritto a uno sconto negli acquisti di libri. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

MITTENTE

COGNOME ____________

NOME ____________

VIA E NUMERO ____________

CAP ______ CITTÀ ____________

PROV. ______ TEL. (______) ____________

GRUPPO EDITORIALE JACKSON
Via Rosellini, 12
20124 Milano

CEDOLA COMMISSIONE LIBRI E GRANDI OPERE

Se desiderate ordinare libri o "Grandi Opere Jackson", utilizzate questa cedola. Compilate gli appositi spazi precisando anche il tipo di pagamento scelto, il vostro nome, cognome e indirizzo. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

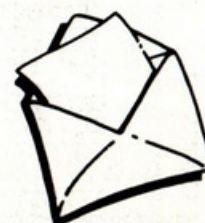
MITTENTE

COGNOME ____________

NOME ____________

VIA E NUMERO ____________

CAP ______ CITTÀ ____________

PROV. ______ TEL. (______) ____________

GRUPPO EDITORIALE JACKSON
Via Rosellini, 12
20124 Milano

CEDOLA INFORMAZIONI E AGGIORNAMENTI

Se desiderate ricevere rapidamente informazioni sui prodotti e attività del Gruppo Editoriale Jackson o acquistare, con formula rateale a sole L. 25.000 mensili e un anticipo di L. 45.000 una "Grande Opera Jackson", barrate le caselle della cedola che vi interessano. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

MITTENTE

COGNOME ____________

NOME ____________

VIA E NUMERO ____________

CAP ______ CITTÀ ____________

PROV. ______ TEL. (______) ____________

GRUPPO EDITORIALE JACKSON
Via Rosellini, 12
20124 Milano

SCUOLA DI ALTE TECNOLOGIE APPLICATE JACKSON S.A.T.A.

CALENDARIO CORSI 1988

Lunedì 29 agosto Inizio corso SPECIALIZZAZIONE IN AUTOMAZIONE INDUSTRIALE E ROBOTICA 175 ore (serale)
Inizio corso MICROPROCESSORI BASE 40 ore (serale)
Lunedì 5 settembre
Inizio corso CONTROLLO E PREVENZIONE DELLE PARTI ELETTRONICHE DALLE SCARICHE ELETTROSTATICHE 24 ore (intensivo)
Inizio corso PROGRAMMAZIONE IN BASIC 40 ore (intensivo)
Inizio corso UNIX, XENIX UTENTI 80 ore (serale)
Inizio corso DBIII PLUS UTENTI 24 ore (serale)
Lunedì 12 settembre
Inizio corso EMC-COMPATIBILITÀ ELETTROMAGNETICA 24 ore (intensivo)
Inizio corso PROGRAMMAZIONE IN PASCAL-TURBOPASCAL 50 ore (intensivo)
Inizio corso CASE-COMPUTER AIDED SOFTWARE ENGINEERING 40 ore (intensivo)
Giovedì 15 settembre
Inizio corso DBIII PLUS PROGRAMMAZIONE 24 ore (serale)
Lunedì 19 settembre
Inizio corso DESK TOP PUBLISHING 40 ore (serale)
Inizio corso TRASDUTTORI, SENSORI, ATTUATORI 20 ore (serale)
Mercoledì 21 settembre
Inizio corso ELETTRONICA DIGITALE 60 ore (intensivo)
Lunedì 26 settembre
Inizio corso PROGRAMMAZIONE IN COBOL 60 ore (intensivo)
Mercoledì 28 settembre
Inizio corso CONTROLLORI LOGICI PROGRAMMABILI 40 ore (serale)
Lunedì 3 ottobre Inizio corso ARCHITETTURA SNA 32 ore (intensivo)
Inizio corso INTEGRAZIONE EDP E TLC NELL'OFFICE AUTOMATION 32 ore (intensivo)
Inizio corso APPLICAZIONI INDUSTRIALI DELLE TECNICHE LASER DI BASSA POTENZA 32 ore (intensivo)
Inizio corso VENTURA 24 ore (intensivo)
Lunedì 10 ottobre
Inizio corso APPARATI E SISTEMI PER LE RETI DI COMPUTER 40 ore (intensivo)
Inizio corso MICROPROCESSORI BASE 40 ore (intensivo)
Inizio corso PAGE MAKER 24 ore (intensivo)
Lunedì 17 ottobre Inizio corso PROGRAMMAZIONE IN C 80 ore (intensivo)
Inizio corso PROGRAMMAZIONE WINDOWS BASE 80 ore (intensivo)
Inizio corso PROCESSORI DI SEGNALE DIGITALE 60 ore (intensivo)
Inizio corso MANUSCRIPT 24 ore (intensivo)
Lunedì 24 ottobre Inizio corso ELEMENTI BASE DI ROBOTICA 20 ore (serale)
Lunedì 7 novembre Inizio corso PC/MS-DOS 24 ore
Inizio corso PIANIFICAZIONE RETICOLARE COL PC 24 ore (intensivo)
Inizio corso OFFICE COMMUNICATION 24 ore (intensivo)
Inizio corso INFOCENTER 32 ore (intensivo)
Inizio corso TECNICHE BASE E SISTEMI PER TRASMISSIONI DATI 80 ore (serale)
Inizio corso MICROPROCESSORI EVOLUTO 40 ore (serale)
Inizio corso INTRODUZIONE ALL'INTELLIGENZA ARTIFICIALE E AI SISTEMI ESPERTI 40 ore (intensivo)
Inizio corso INFORMIX/SQL 50 ore (serale)
Inizio corso ARCHITETTURA OS/2 40 ore (intensivo)
Inizio corso MICROPROCESSORI A 16 BIT 60 ore (intensivo)
Inizio corso AFFIDABILITÀ DEI CIRCUITI E DEI COMPONENTI ELETTRONICI 24 ore (intensivo)
Mercoledì 9 novembre Inizio corso WORD 24 ore
Lunedì 14 novembre
Inizio corso RETI DI COMUNICAZIONE NELLA FABBRICA AUTOMATIZZATA 20 ore (serale)
Inizio corso MODELLI PREVISIONALI COL PC 24 ore (intensivo)
Inizio corso USO DEL PC NELL'AREA PRODUZIONE 24 ore (intensivo)
Inizio corso PROGETTAZIONE DEI MODERNI CIRCUITI STAMPATI 24 ore (intensivo)
Lunedì 21 novembre
Inizio corso RETI A COMMUTAZIONE DI PACCHETTO 40 ore (intensivo)
Inizio corso PROGRAMMAZIONE IN LISP 40 ore (intensivo)
Inizio corso MODELLI DECISIONALI COL PC 24 ore (intensivo)
Inizio corso USO DEL PC NELL'AREA MARKETING 24 ore (intensivo)
Lunedì 28 novembre Inizio corso MULTIPLAN 24 ore
Inizio corso PROGRAMMAZIONE IN PROLOG 40 ore (intensivo)
Inizio corso AUTO-CAD 32 ore (serale)
Inizio corso SERVIZI A VALORE AGGIUNTO SULLE RETI X25 24 ore (intensivo)
Mercoledì 30 novembre Inizio corso LOTUS 1-2-3 24 ore
Lunedì 12 dicembre
Seminario con WORK-SHOP SUI LINGUAGGI DELLA IV GENERAZIONE 24 ore (intensivo)
Inizio corso IL MODELLO OSI 32 ore (intensivo)
Inizio corso SYMPHONY 40 ore (intensivo)
Inizio corso OTTIMIZZAZIONE E DEBUGGING "C" 40 ore (intensivo)
Inizio corso MICROPROCESSORI EVOLUTO 40 ore (intensivo)

SCUOLA DI ALTE TECNOLOGIE APPLICATE

GRUPPO EDITORIALE JACKSON — DIVISIONE FORMAZIONE

S.A.T.A.

Per le modalità di iscrizione e richiesta di programmi dettagliati, telefonare alla DIVISIONE FORMAZIONE E PRODOTTI PER LA DIDATTICA del Gruppo Editoriale Jackson Via Imperia 2 Milano Telefono 8467500

N.B. Per abbonamenti all'estero le tariffe dovranno essere raddoppiate. Non è prevista la spedizione via aerea.

SÌ DESIDERO ABBONARMI ALLE SEGUENTI RIVISTE JACKSON:

Rivista	Numeri	Prezzo
☐ INDUSTRIA OGGI	numeri 10	L. 41.000
☐ ELETTRONICA OGGI	numeri 20	L. 79.000
☐ AUTOMAZIONE OGGI	numeri 20	L. 78.000
☐ EO NEWS	numeri 40	L. 79.500
☐ WATT	numeri 11	L. 35.500
☐ TRASMISSIONE DATI E TELECOMUNICAZIONI	numeri 11	L. 44.000
☐ VIDEOTEL MAGAZINE	numeri 6	L. 20.000
☐ INFORMATICA OGGI	numeri 11	L. 40.000
☐ INFORMATICA OGGI SETTIMANALE	numeri 40	L. 80.000
☐ LAB NEW	numeri 8	L. 36.000
☐ COMPUTER GRAFICA & APPLICAZIONI	numeri 6	L. 30.000
☐ PC WORLD MAGAZINE	numeri 11	L. 44.000
☐ PCW Magazine + PC Floppy	numeri 11	L. 105.000
☐ BIT	numeri 11	L. 43.000
☐ SUPERCOMMODORE 64 & 128 (con cassetta)	numeri 11	L. 66.000
☐ SUPERCOMMODORE 64 & 128 (con disco)	numeri 11	L. 105.000
☐ COMMODORE PROFESSIONAL	numeri 10	L. 48.000
☐ OLIVETTI PRODEST USER	numeri 6	L. 20.000
☐ LA RIVISTA DI ATARI	numeri 6	L. 24.000
☐ COMPUSCUOLA	numeri 10	L. 32.000
☐ FARE ELETTRONICA	numeri 12	L. 39.000
☐ STRUMENTI MUSICALI	numeri 11	L. 35.000
☐ NAUTICAL QUARTERLY	numeri 4	L. 70.000
☐ AUTOMOBILE QUARTERLY	numeri 4	L. 69.500

☐ L'abbonamento dovrà decorrere dal mese di ______

MODALITÀ DI PAGAMENTO

Per il pagamento ☐ Allego assegno n. ______ di L. ______

Banca ______

☐ Ho effettuato versamento di L. ______ sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson - Milano e allego fotocopia della ricevuta.

☐ Ho effettuato versamento di L. ______ tramite vaglia postale o telegrafico e allego fotocopia della ricevuta.

☐ Vi autorizzo ad addebitare l'importo di L. ______ sulla carta di credito.

☐ VISA ☐ AMERICAN EXPRESS ☐ DINERS CLUB ☐ CARTA SI

N. ______ Data di scadenza ______

Data ______ Firma ______

SÌ INVIATEMI LE SEGUENTI "GRANDI OPERE JACKSON"

Codice	Q.tà	Prezzo	Codice	Q.tà	Prezzo

Ordine minimo L. 30.000 + L. 3.500 per contributo fisso spese di spedizione

☐ **Sono abbonato** alla seguente rivista Jackson: ______
e ho quindi diritto allo sconto del **10%**.

SÌ INVIATEMI I VOLUMI SOTTOELENCATI:

Volume	Prezzo
☐ **EI - Enciclopedia di Elettronica e Informatica** 10 volumi	L. 476.000
☐ **SOFTWARE** 5 volumi	L. 236.000
☐ **DEI - Dizionario di Elettronica e Informatica** 10 volumi	L. 276.000
☐ **Enciclopedia Monografica di Elettronica e Informatica** 2 volumi	L. 116.000
☐ **ABC Personal Computer** 4 volumi	L. 136.000
☐ **VIDEOBASIC MSX** 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC SPECTRUM** 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** C64/C128/64PC 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** C64/C128/64PC 20 lezioni + 10 floppy	L. 176.000
☐ **VIDEOBASIC** C16/PUS4 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** VIC20 20 lezioni + 20 cassette	L. 176.000
☐ **CORSO DI GRAFICA** C64/C128/64PC 10 lez. + 10 cass.	L. 96.000
☐ **A SCUOLA DI SCACCHI** C64/C128/64PC 10 lez. + 10 cass.	L. 96.000
☐ **7 NOTE BIT** C64/C128/64PC 15 lezioni + 15 cassette	L. 156.000
☐ **LABORATORIO DI ELETTRONICA** 5 vol. (disp. da giugno 88)	L. 236.000
☐ **BYTES** 6 volumi (disp. da giugno 1988)	L. 276.000

☐ Allego assegno n. ______ di L. ______
della Banca ______

☐ Ho effettuato il pagamento di L. ______ a mezzo:
☐ vaglia postale ☐ vaglia telegrafico ☐ versamento sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson SpA Milano e allego fotocopia della ricevuta.

☐ Pagherò al postino l'importo di L. ______ al ricevimento dell'opera.

☐ Vi autorizzo ad addebitare l'importo di L. ______ sulla carta di credito: ☐ Visa ☐ American Express ☐ Diners Club ☐ Carta SI
conto n. ______ data di scadenza ______

☐ Richiedo l'emissione della fattura (formula riservata alle aziende) e comunico il numero di Partita IVA ______

DATA ______ FIRMA ______

CEDOLA INFORMAZIONI E AGGIORNAMENTI

☐ Desidero riceve informazioni sulle **riviste** JACKSON
☐ Desidero ricevere una copia saggio della **rivista** JACKSON ______
(indicare titolo)
☐ Desidero ricevere il **catalogo libri tecnici** JACKSON
☐ Desidero ricevere il **catalogo libri scolastici** JACKSON
☐ Desidero ricevere il **catalogo grandi opere** JACKSON e una copia saggio ______
(indicare titolo)
☐ Desidero ricevere informazioni per l'acquisto in **forma rateale delle Grandi Opere** JACSKON e una copia saggio ______
(indicare titolo)
☐ Desidero ricevere il **catalogo occasioni** JACKSON

Desidero ricevere maggiori informazioni sulla **Divisione Formazione e Prodotti per la Didattica**
☐ Area ELETTRONICA E MICROPROCESSORI
☐ Area TELECOMUNICAZIONI E TELEMATICA
☐ Area AUTOMAZIONE INDUSTRIALE E ROBOTICA
☐ Area INFORMATICA E INTELLIGENZA ARTIFICIALE
☐ Area ALTE TECNOLOGIE SPECIALI
☐ Laboratorio di INFORMATICA
☐ Laboratorio di TELEMATICA
☐ Laboratorio di OFFICE AUTOMATION
☐ Laboratorio di ELETTRONICA
☐ Laboratorio di MICROPROCESSORI
☐ Laboratorio di AUTOMAZIONE INDUSTRIALE

COMMODORE professional
La rivista specializzata per gli utenti Commodore
AMIGA
E finalmente musica
MODULA-2 e Amiga
Dietro le quinte del Filing System
C 128
Funzioni matematiche
Questionario
Rotazione
Punti e linee
C 64
Il linguaggio del DOS
STRUMENTI MUSICALI
Il mensile per i professionisti della musica: audiotest, rassegne, computer music, servizi, interviste e recensioni delle ultime novità discografiche.
strumenti MUSICALI
MENSILE DI STRUMENTI E INFORMATICA MUSICALE
N. 97 MARZO 1988 L. 4.000 Frs 6,00
WENDY E LISA
PHILIP GLASS
MIDI BOX (3ª parte)
COMMODORE PROFESSIONAL
La rivista professionale per gli utenti di Commodore Amiga, C128 e C64.
OGNI MESE IN EDICOLA
BIT
L. 5000 Frs. 7.50
ANNO 11 N. 91 FEBBRAIO 1988
HERON, LA CONTINUITÀ
SPECIALE CD-ROM & WORD PROCESSOR II
APPLE & GO! Il disco sistema
VETRINA Hard disk SCSI Mactronics Il software per Amiga
ON-LINE Il modem Flycom
SOFTEST Freelance, by Lotus
JACKSON
BIT
La prima rivista europea, la più famosa e autorevole in Italia, di personal, home, business computer, software e accessori.